Part II: Probability

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 chil-

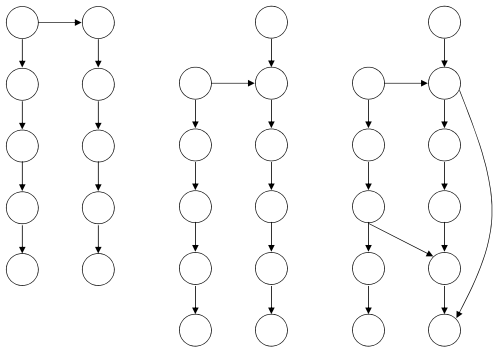


Figure 1. From left to right: a tree, a polytree, and a multiply–connected network.

dren were said to quantify the causal 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 algo-

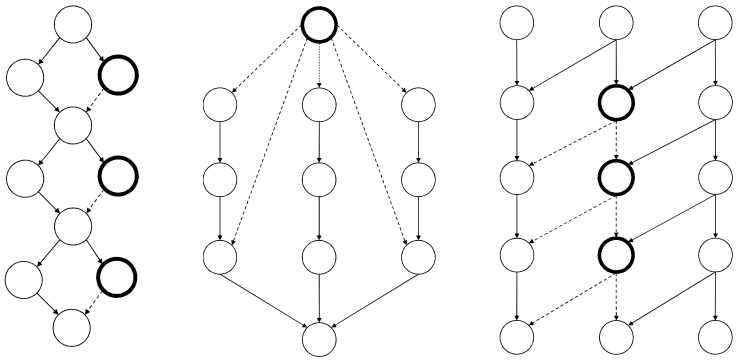


Figure 2. Networks and corresponding loop-cutsets (bold circles).

rithm [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.

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 al-

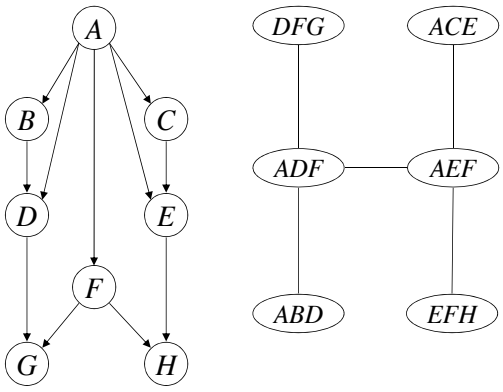


Figure 3. A Bayesian network structure and its corresponding jointree (tree of clusters).

ternative 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 until 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

[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

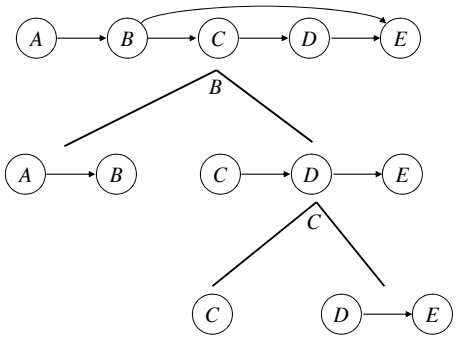


Figure 4. Decomposing a Bayesian network by conditioning on variable B and then on variable C.

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 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

connecting every pair of nodes that share a child in the DAG and then dropping the directionality of all edges.

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

[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

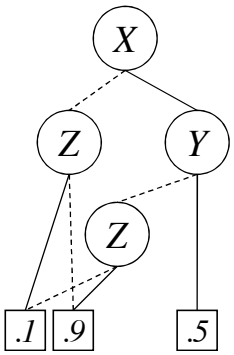


Figure 5. A factor over binary variables X, Y, Z with a tabular representation (left) and an ADD representation (right).

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).

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 subcomputations 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

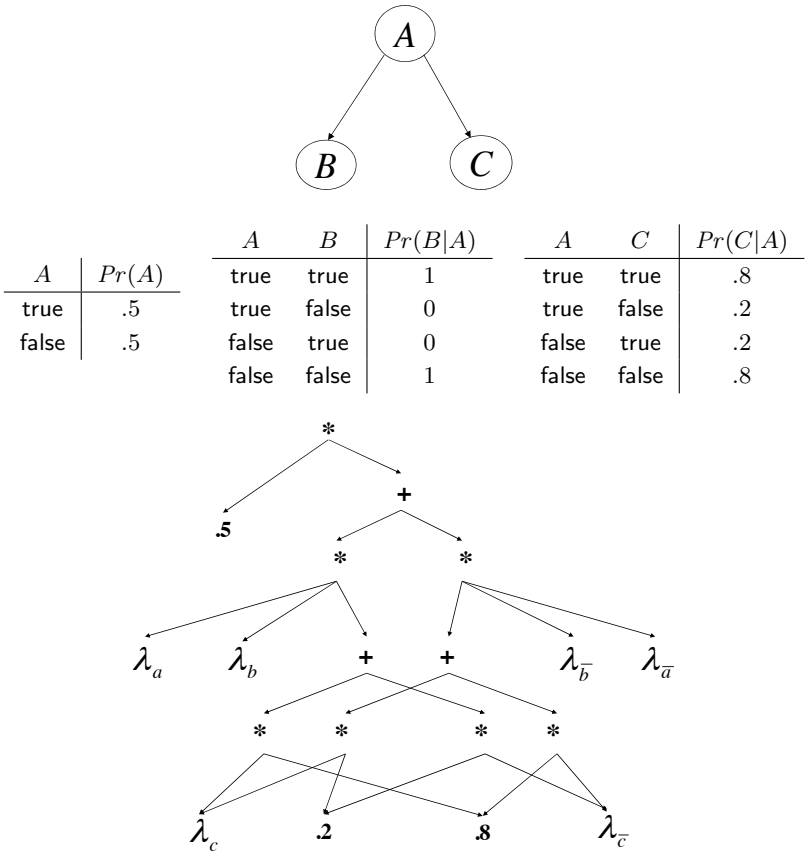


Figure 6. A Bayesian network and a corresponding arithmetic circuit.

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

[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.

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, 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 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

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

[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.

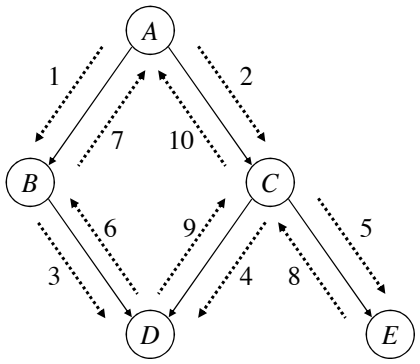


Figure 7. A Bayesian network annotated with an ordering of LBP messages (leading to a sequential message passing schedule).

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].

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 induces 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 computations 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.

8

Graphical Models of the Visual Cortex

Thomas Dean

1 Pivotal Encounters with Judea

Post graduate school, three chance encounters reshaped my academic career, and all three involved Judea Pearl directly or otherwise. The first encounter was meeting Judea on a visit to the UCLA campus at a time when I was developing what I called temporal Bayesian networks and would later be called *dynamic belief networks* (an unfortunate choice of names for reasons I'll get to shortly). Judea was writing his book *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* [1988] and his enthusiasm for the subject matter was positively infectious. I determined from that meeting that I was clueless about all things probabilistic and proceeded to read each of Judea's latest papers on Bayesian networks multiple times, gaining an initial understanding of joint and marginal probabilities, conditional independence, etc. In those days, a thorough grounding in probability and statistics was rarely encouraged for graduate students working in artificial intelligence.

The second encounter was with Michael Jordan at a conference where he asked me a question that I was at a loss to answer and made it clear to me that I didn't really understand Bayesian probability theory at all, despite what I'd picked up from Judea's papers. My reaction to that encounter was to read Judea's book cover to cover and discover the work of I.J. Good. Despite being a math major and having met I.J. Good at Virginia Tech where I was an undergraduate and Good was a professor of statistics, I never took a course in probability or statistics. My embarrassment at being flummoxed by Mike's question forced me to initiate a crash course in probability theory based on the textbooks of Morris DeGroot [1970, 1986]. I didn't recognize it at the time, but Judea, Mike and like-minded researchers in central areas of artificial intelligence were in the vanguard of those changing the landscape of our discipline.

The third encounter was with David Mumford when our paths crossed in the midst of a tenure hearing at Brown University and David told me of his work on models of the visual cortex. I read David's paper with Tai Sing Lee [2003] as well as David's earlier related work [1991, 1992] and naïvely set out to implement their ideas as a probabilistic graphical model [Dean 2005]. Indeed, I wanted to extend their work since it did not address the representation of time passing, and I was interested in building a model that dealt with how a robot might make sense of its observations as it explores its environment.

Moreover, the theory makes no mention of how a robot might learn such a model, and, from years of working with robots, I was convinced that building a model by hand would turn out to be a lot of work and very likely prove to be unsuccessful. Here it was Judea's graphical-models perspective that, initially, made it easy for me to think about David's work, and, later, extend it. I also came to appreciate the relevance of Judea's work on causality and, in particular, the role of intervention in thinking about how biological systems engage the world to resolve perceptual ambiguity.

This chapter concerns how probabilistic graphical models might be used to model the visual cortex, and how the challenges faced in developing such models suggest areas where current theory falls short and might be extended. A graphical model is a useful formalism for compactly describing a joint probability distribution characterized by very large number of random variables. We are taking what is known about the anatomy and physiology of the primate visual cortex and attempting to apply that knowledge to construct probabilistic graphical models that we can ultimately use to simulate some functions of primate vision. It may be that the resulting probabilistic model also captures some important characteristics of individual neurons or their ensembles. For practical purposes, this need not be the case, though clearly we believe there are potential advantages to incorporating some lessons from biology into our models. Graphical models also suggest, but do not dictate, how one might use such a model along with various algorithms and computing hardware to perform inference and thereby carry out practical simulations. It is this latter use of graphical models that we refer to when we talk about implementing a model of the visual cortex.

2 Primate Visual Cortex

Visual information processing starts in the retina and is routed via the optic tract to the lateral geniculate nuclei (LGN) and then on to the striate cortex also known as visual area one (V1) located in the occipital lobe at the rear of the cortex. There are two primary visual pathways in the primate cortex: The ventral pathway leads from the occipital lobe into the temporal lobe where association areas in the inferotemporal cortex combine visual information with information originating from the auditory cortex. The dorsal pathway leads from the occipital to the parietal lobe which, among other functions, facilitates navigation and manipulation by integrating visual, tactile and proprioceptive signals to provide our spatial sense and perception of shape.

It is only in the earliest portion of these pathways that we have any reasonably accurate understanding of how visual information is processed, and even in the very earliest areas, the striate cortex, our understanding is spotty and subject to debate. It seems that cells in V1 are mapped to cells in the retina so as to preserve spatial relationships, and are tuned to respond to stimuli that appear roughly like oriented bars. Hubel and Wiesel's research on macaque monkeys provides evidence for and

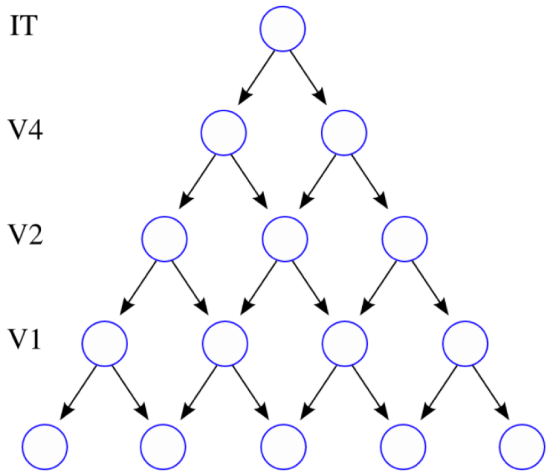


Figure 1. A simple hierarchical model of the ventral visual pathway.

subsequent studies confirm the latter characterization of function in primates [1962, 1968]. That said, there is still a good deal that we don't know about visual processing in V1 [Olshausen and Field 2005], and our understanding gets murkier as we progress along these pathways.

It is said that the ventral pathway is responsible for identifying "what" objects we see, and the dorsal for identifying "where" in our visual field these objects and their various parts are located and "how" we might interact with them by grasping, avoiding, etc. This is sufficiently vague that, given our current understanding of visual processing, it is probably a useful rule of thumb. However, there is ample evidence [Konen and Kastner 2008] to suggest that the "what", "where" and "how" are commingled via a myriad of connections and it is misleading to think of visual processing as a pipeline that leads in pure feed-forward fashion from simple features that subtend small portions of the visual field to more complex features that subtend greater and greater spatial (and temporal) extent. Indeed, there is no conclusive evidence that the brain is organized as a hierarchy of features, despite this being an elegant and comforting hypothesis to entertain.

3 Static Graphical Models

Figure 1 depicts a simple graphical model of the ventral visual pathway, where the nodes in the bottom layer are meant to model retinal ganglion cells. Nodes in the second layer model cells in the striate cortex which is also known as Brodmann's area 17 or V1. The next layer corresponds to Brodmann's area 18 or V2 which is responsive to somewhat more complex patterns than V1. The penultimate layer encodes V4 which is tuned to object features of intermediate complexity, and the

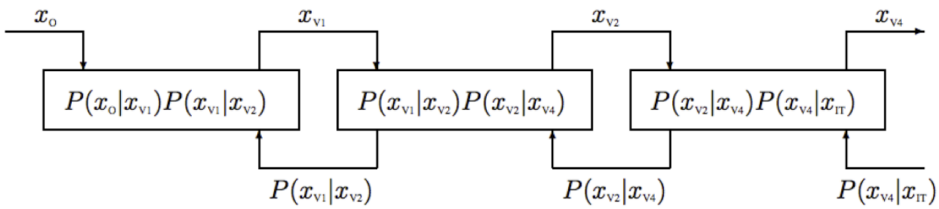


Figure 2. A schematic of the hierarchical Bayesian framework proposed by Lee and Mumford [2003]. The regions of the visual cortex are linked together in a Markov chain. The activity in the ith region is influenced by bottom-up feed-forward data x_{i-1} and top-down probabilistic priors $P(x_i|x_{i+1})$ representing feedback from region $i+1$. The Markov property plays an important computational role by allowing units to depend only on their immediate neighbors in the Markov chain.

final layer represents inferotemporal cortex or IT which apparently responds to complex shapes. To get a more realistic picture, imagine that each layer represents a two-dimensional map with correspondence to the surface of the retina. Let's consider several ways in which this simple model falls short of the mark.

The graph in Figure 1 has no edges between nodes in the same layer. We know adjacent cells within each of V1, V2, V4 and IT communicate via many lateral connections. Given such connectivity, it would seem that two nodes representing adjacent cells in a layer are dependent in a statistical sense. However, the model as it stands has the property that any two nodes in a given layer are independent of one another when conditioned on the nodes in the layer immediately above. It turns out that it is very difficult to capture complex spatial relationships among adjacent regions of images using a graphical model having the intra-layer conditional independence implicit in model shown in Figure 1.

How might we capture the statistical properties of adjacent cells in the same cortical layer in a graphical model? First, note that our graphical model, while more complex than a tree, is simpler than an arbitrary directed graph being acyclic. Unfortunately, it is difficult to devise a plausible scheme to connect vertices within layers using directed edges and avoid cycles in the graph, and so we won't even attempt this. One possibility is that we model each layer as a Markov random field with the connectivity of a regular grid, but retain the directional edges between layers. The advantage of this approach is that the graph as a whole is a Markov chain and it is relatively simple to write down the joint distribution. Using the chain rule and the assumption made explicit in the graph shown in Figure 1 that each variable in the sequence $(x_{\mathrm{O}}, x_{\mathrm{V1}}, x_{\mathrm{V2}}, x_{\mathrm{V4}}, x_{\mathrm{IT}})$ is independent of the other variables given its immediate neighbors in the sequence, we write the equation relating the

one retinal and three cortical regions as

$$P(x_{\mathrm{O}}, x_{\mathrm{V1}}, x_{\mathrm{V2}}, x_{\mathrm{V4}}, x_{\mathrm{IT}}) = P(x_{\mathrm{O}}, x_{\mathrm{V1}})P(x_{\mathrm{V1}}, x_{\mathrm{V2}})P(x_{\mathrm{V2}}, x_{\mathrm{V4}})P(x_{\mathrm{V4}}, x_{\mathrm{IT}})P(x_{\mathrm{IT}})$$

where x_{O} represents the retinal or *observation* layer. Moreover, we know that, although the edges all point in the same direction, information flows both ways in the hierarchy via Bayes rule (see Figure 2).

Despite the apparent simplicity when we collapse each layer of variables into a single, joint variable, exact inference in such a model is intractable. One might imagine, however, using a variant of the forward-backward algorithm to approximate the joint distribution over all variables. Such an algorithm might work one layer at a time, by isolating each layer in turn, performing an approximation on the isolated Markov network using Gibbs sampling or mean-field approximation, propagating the result either forward or backward and repeating until convergence. Simon Osindero and Geoff Hinton [2008] experimented with just such a model and demonstrated that it works reasonably well at capturing the statistics of patches of natural images.

One major problem with such a graphical model as a model of the visual cortex is that the Markov property of the collapsed-layer simplification fails to capture the inter-layer dependencies implied by the connections observed in the visual cortex. In the cortex as in the rest of the brain, connections correspond to the dendritic branches of one neuron connected at a synaptic cleft to the axonal trunk of a second neuron. We are reasonably comfortable modeling such a *cellular edge* as an edge in a probabilistic graphical model because for every cellular edge running forward along the visual pathways starting from V1 there is likely at least one and probably quite a few cellular edges leading backward along the visual pathways. Not only do these backward-pointing cellular edges far outnumber the forward-pointing ones, they also pay no heed to the Markov property, typically spanning several layers of our erstwhile simple hierarchy. Jin and Geman [2006] address this very problem in their hierarchical, compositional model, but at a considerable computational price. Advances in the development of adaptive Monte Carlo Markov chain (MCMC) algorithms may make inference in such graphical models more practical, but, for the time being, inference on graphical models of a size comparable to the number of neurons in the visual cortex remains out of reach.

4 Temporal Relationships

Each neuron in the visual cortex indirectly receives input from some, typically contiguous, region of retinal ganglion cells. This region is called the neuron's *receptive field*. By introducing lags and thereby retaining traces of earlier stimuli, a neuron can be said to have a receptive field that spans both space and time — it has a *spatiotemporal* receptive field. A large fraction of the cells in visual cortex and V1 in particular have spatiotemporal receptive fields. Humans, like most animals, are very attentive to motion and routinely exploit motion to resolve visual ambiguity,

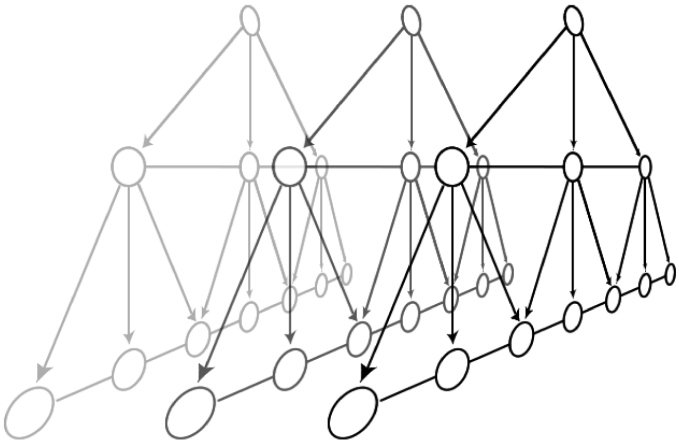


Figure 3. A cartoon of the hierarchical hidden Markov model described in [Dean 2006] showing the same basic structure as in Figure 1, but with additional undirected, intra-layer edges, and replicated for some number of time steps to form a hierarchy of hidden Markov models, so that nodes in a each layer represent different spatial and temporal extent. Not shown, but present in the full model, are edges that span the temporal slices thus modeling neural circuits that have spatiotemporal receptive fields.

and, generally, deal with the four-dimensional, space-time continuum in which we live.

Figure 3 depicts the obvious temporal analog of Figure 1. Dean [2006] presents a model of visual cortex based on the Hierarchical Hidden Markov Model of Fine *et al* [1998]. In the model described in [2006], nodes have edges that span both nodes within the layers of individual time slices and nodes that reside within the same layer of adjacent time slices.

The ability to represent the passage of time is clearly important in enabling us to make predictions and plan our actions, but it also allows us to expedite learning. The cortex evolved to take advantage of *temporal coherence*, the property that, for the most part, the appearance of the objects present in our visual field, animate or otherwise, do not change a great deal from one instant to the next. We can exploit this property to learn about the stable features of our environment. Földiák [Földiák 1991] describes a biologically plausible theory of how neural circuits might exploit temporal coherence to learn useful features by looking for signals that change slowly over time. Wiskott and Sejnowski [2002], Hyvärinen*et al* [2003], George and Hawkins [2005], Dean [2006] and others have proposed various algorithms for improving on this same basic idea.

Using proximity in space and time to group similar visual features was recognized early on by the Gestalt psychologists, and it is one of many characteristics of our

visual experience that biology has evolved to exploit to its advantage. However, in this chapter, I want to explore a different facet of how we make sense of and, in some cases, take advantage of spatial and temporal structure to survive and thrive, and how these aspects of our environment offer new challenges for applying graphical models.

5 Dynamic Graphical Models

Whether called temporal Bayesian networks [Dean and Wellman 1991] or dynamic Bayesian networks [Russell and Norvig 2003], these graphical models are designed to model properties of our environment that change over time and the events that precipitate those changes. The networks themselves are not dynamic: the numbers of nodes and edges, and the distributions that quantify the dependencies among the random variables that correspond to the nodes are fixed. At first blush, graphical models may seem a poor choice to model the neural substrate of the visual cortex which is anything but static. However, while the graph that comprises a graphical model is fixed, a graphical model can be used to represent processes that are highly dynamic, and contingent on the assignments to observed variables in the model. In the remainder of this section, we describe characteristics of the visual system that challenge our efforts to model the underlying processes required to simulate primate vision well enough to perform such tasks such as object recognition and robot navigation.

The retina and the muscles that control the shape of the lens and the position of the eyes relative to one another and the head comprise a complex system for acquiring and processing visual information. A mosaic of photoreceptors activate several layers of cells, the final layer of which consists of retinal ganglion cells whose axons comprise the optic nerve. This multi-layer extension of the brain performs a range of complex computations ranging from light-dark adaptation to local contrast normalization [Brady and Field 2000]. The information transmitted along the optic tract is already the product of significant computational processing.

Visual information is *retinotopically* mapped from the retinal surface to area V1 so as to preserve the spatial relationships among patches on the retina that comprise the receptive fields of V1 cells. These retinotopic mappings are primarily sorted out *in utero*, but the organization of the visual cortex continues to evolve significantly throughout development — this is particularly apparent when children are learning to read [Dehaene 2009]. Retinotopic maps in areas beyond V1 are more complicated and appear to serve purposes that relate to visual tasks, *e.g.*, the map in V2 anatomically divides the tissue responsible for processing the upper and lower parts of the visual fields. These retinotopic maps, particularly those in area V1, have led some computer-vision researchers to imagine that early visual processing proceeds via transformations on regular grid-like structures with cells analogous to pixels.

The fact is that our eyes, head, and the objects that we perceive are constantly

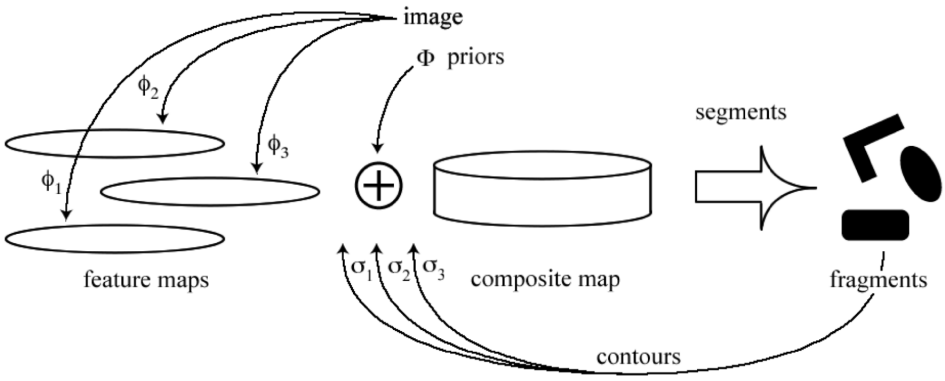


Figure 4. This graphic depicts a variant of the process of segmentation as proposed by Tenenbaum and Barrow [1977] in which the low-level features ϕ_i are used to construct feature maps that are combined with priors Φ derived from context to form composite maps which produce candidate fragments that suggest the boundaries or contours σ_i of objects which in turn guide subsequent interpretation and assignment of fragments to objects.

in flux. Even with our head fixed while viewing a still image, our eyes quickly jump or *saccade* up to 90° of visual angle several times a second, and perform much smaller adjustments or *microsaccades* at around 30–50 Hertz. A two-week-old baby can already track objects by *smooth pursuit*, thereby keeping an object centered in the *fovea* which is the central, high-acuity and high-color-sensitivity portion of the retina. Both smooth pursuit and (macro) saccades are driven by attentional mechanisms which combine a bottom-up, small-image-patch, data-driven component with a top-down, whole-image-gist, prior-knowledge component. The important point here is that how we perceive the visual world has little resemblance — beyond the earliest processing stages — to a sequence of orderly transformations performed on a regular grid, and has everything to do with assembling a puzzle out of fragmentary glimpses snatched as our gaze quickly shifts relative to the frames of reference of our head, our body and the ground on which we stand.

Even if we concentrate on the hundred milliseconds or so during which the fovea remains focused on a small patch of a still image between saccades, the representation of this process as a graphical model becomes complicated as we abstract from the "pixel" level. Figure 4 depicts a process whereby the responses of low-level feature detectors are used to construct feature maps. Typically, the feature detectors report information about intensity, color, texture, etc. These maps are combined so that every location in an image is summarized by a vector of features. In bottom-up segmentation, such summaries alone are used to aggregate locations into segments that correspond to object surfaces or at least respect object boundaries. Tenenbaum

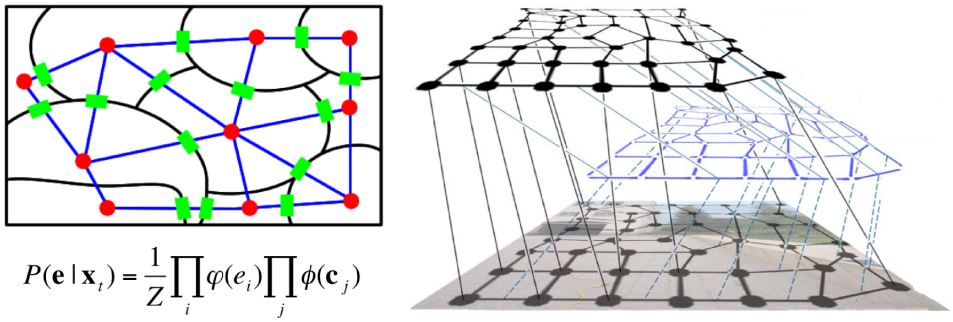


Figure 5. The graphic in the upper left depicts an over-segmentation of an image superimposed with a graph whose nodes (depicted as circles) correspond to segments or *superpixels* and whose edges (marked with rectangular boxes) correspond to boundaries between segments. The graphic on the right (adapted from Saxena *et al* [2007]) shows a graphical model in the form of multi-layer factor graph where nodes in the lowest layer correspond to superpixels, those in the second layer to boundary classes and those in the top to object surfaces.

and Barrow [1977] suggested that this low-level information has to be supplemented by *prior* knowledge, which provides a more global context in which to interpret the low-level information, and combined in an iterative of process of agglomeration.

This process has been realized in a graphical model format using variants of factor graphs [Saxena, Chung, and Ng 2007] and conditional random fields [Hoiem, Efros, and Hebert 2007] and hierarchical Dirichlet-process hidden-Markov trees [Kivinen, Sudderth, and Jordan 2007]. Figure 5 characterizes the basic structure of the algorithm adopted by Saxena *et al* [2007] and by Hoiem *et al* [2007]. First, the raw image is divided into *superpixels* — regions of pixels homogeneous in their intensity, color or texture — which correspond to the segments obtained from an over-segmentation of the image that is assumed to respect image boundaries. Next, the superpixels are used to construct a graphical model consisting of nodes corresponding superpixels, information that pertains to their status as object (occlusion) boundaries, and their relationships *vis a vis* their contribution to the same objects. Inference serves to identify boundaries or the absence thereof, some superpixels can be merged, and the process repeated until no further merging is possible.

In the implementation of this process, the topology of the graphical model is generated on an image-by-image basis, and the iterative refinement process requires adjustments to the size and connectivity of the graph that are particular to the boundaries of object surfaces in the image being observed. Such a process can be implemented within a fixed-graph structure but the model of a dynamically changing graph is simple to implement and apply recursively. One advantage, however, of a graph in which the nodes in a fixed grid of nodes are dynamically assigned to

segments as part of inference is that this model is potentially more elegant, and even biologically plausible, in that the recursive process might be represented as a single hierarchical graphical model allowing inference over the entire graph, rather than over sequences of ever more refined graphs.

The above discussion of segmentation is but one example in which nodes in a graphical model might serve as *generic* variables that are bound as required by circumstances. But perhaps this view is short sighted; why not just assume that there are enough nodes that every possible (visual) concept corresponds to a unique combination of existing nodes. In this view, visual interpretation is just mapping visual stimuli to the closest visual "memory". Given the combinatorics, the only way this could be accomplished is to use a hierarchy of features whose base layer consists of small image fragments at many different spatial scales, and all subsequent layers consist of compositions of features at layers lower in the hierarchy [Bienenstock and Geman 1995; Ullman and Soloviev 1999; Ullman, Vidal-Naquet, and Sali 2002]. This view accords well with the idea that most visual stimuli are not determined to be novel and, hence, we construct our reality from bits and pieces of existing memories [Hoffman 1998]. Our visual memories are so extensive that we can almost always create a plausible interpretation by recycling old memories. It may be that in some aspects of cognition we have to employ generic neural structures to perform the analog of binding variables, but for much of visual intelligence this may not be necessary given a large enough memory of reusable fragments. Which raises the question of how we might implement a graphical model that has anywhere near the capacity of the visual cortex.

6 Distributed Processing at Cortex Scale

The cortex consists of a layered sheet with a more-or-less uniform cellular structure. Neuroanatomists have identified what are called *columns* corresponding to groups of local cells running perpendicular to the cortical surface. Vernon Mountcastle [2003] writes "The basic unit of cortical operation is the *minicolumn* [...] [containing] on the order of 80–100 neurons [...] The minicolumn measures of the order of 40-50μ in transverse diameter, separated from adjacent minicolumns by vertical cell-sparse zones which vary in size in different cortical areas." These minicolumns are then grouped into cortical columns which "are formed by the binding together of many minicolumns by common input and short-range horizontal connections."

If we take the cortical column — not the minicolumn — as our basic computational module as in [Anderson and Sutton 1997], then the gross structure of the neocortex consists of a dense mat of inter-columnar connections in the outer-most layer of the cortex and another web of connections at the base of the columns. The inter-columnar connectivity is relatively sparse (something on the order of 10^{15} connections spanning approximately 10^{11} neurons) and there is evidence [Sporns and Zwi 2004] to suggest that the induced inter-columnar connection graph exhibits the properties of a *small-world graph* [Newman, Watts, and Strogatz 2002]. In partic-

ular, evidence suggests the inter-columnar connection graph has low diameter (the length of the longest shortest path separating a pair of vertices in the graph) thereby enabling relatively low-latency communication between any two cortical columns.

It is estimated that there are about a quarter of a billion neurons in the primary visual cortex — think V1 through V4 — counting both hemispheres, but probably only around a million or so cortical columns. If we could roughly model each cortical column with a handful of random variables, then it is at least conceivable that we could implement a graphical model of early vision.

To actually implement a graphical model of visual cortex using current technology, the computations would have to be distributed over many machines. Training such a model might not take as long as raising a child, but it could take many days — if not years — using the current computer technology, and, once trained, we presumably would like to apply the learned model for much longer. Given such extended intervals of training and application, since the mean-time-til-failure for the commodity-hardware-plus-software that comprise most distributed processing clusters is relatively short, we would have to allow for some means of periodically saving local state in the form of the parameters quantifying the model.

The data centers that power the search engines of Google, Yahoo! and Microsoft are the best bet that we currently have for such massive and long-lived computations. Software developed to run applications on such large server farms already have tools that could opportunistically allocate resources to modify the structure of graphical model in an analog of neurogenesis. These systems are also resistant to both software and equipment failures and capable of reallocating resources in the aftermath of catastrophic failure to mimic neural plasticity in the face of cell death.

In their current configuration, industrial data centers may not be well suited to the full range of human visual processing. Portions of the network that handle very early visual processing will undoubtedly require shorter latencies than is typical in such server farms, even among machines on the same rack connected with high-speed Ethernet. Riesenhuber and Poggio [1999] use the term *immediate recognition* to refer to object recognition and scene categorization that occur in the first 100-200ms or so from the onset of the stimuli. In that short span of time — less than the time it takes for a typical saccade, we do an incredibly accurate job of recognizing objects and inferring the gist of a scene. The timing suggests that only a few steps of neural processing are involved in this form of recognition, assuming 10–20ms per synaptic transmission, though given the small diameter of the inter-columnar connection graph, many millions of neurons are likely involved in the processing. It would seem that at least the earliest stages of visual processing will have to be carried out in architectures capable of performing an enormous number of computations involving a large amount of state — corresponding to existing pattern memory — with very low latencies among the processing units. Hybrid architectures that combine conventional processors with co-processors that provide fast matrix-matrix and matrix-vector operations will likely be necessary to handle

even a single video stream in real-time.

Geoff Hinton [2005, 2006] has suggested that a single learning rule and a relatively simple layer-by-layer method of training suffices for learning invariant features in text, images, sound and even video. Yoshua Bengio, Yann LeCun and others have also had success with such models [LeCun and Bengio 1995; Bengio, Lamblin, Popovici, and Larochelle 2007; Ranzato, Boureau, and LeCun 2007]. Hyvärinen *et al* [2003], Bruno Olshausen and Charles Cadieu [2007, 2008], Dean *et al* [2009] and others have developed hierarchical generative models to learn sparse codes resembling the responses of neurons in the medial temporal cortex of the dorsal pathway. In each case, the relevant computations can be most easily characterized in terms of linear algebra and implemented using fast vector-matrix operations best carried out on a single machine with lots of memory and many cores (graphics processors are particularly well suited to this sort of computation).

A more vexing problem concerns how we might efficiently implement any of the current models of Hebbian learning in an architecture that spans tens of thousands of machines and incurs latencies measured in terms of milliseconds. Using super computers at the national labs, Eugene Izhikevich and Gerald Edelman [2008] have performed spike-level simulations of millions of so-called *leaky integrate and fire* neurons with fixed, static connections to study the dynamics of learning in such ensembles. Paul Rhodes and his team of researchers at Evolved Machines have taken things a step further in implementing a model that allows for the dynamic creation of edges by simulating dendritic tree growth and the chemical gradients that serve to implement Hebbian learning. In each case, the basic model for a neuron is incredibly simple when compared to the real biology. It is not at all surprising that Henry Markram and his colleagues at EPFL (Ecole Polytechnique Fédérale de Lausanne) require a powerful supercomputer to simulate even a single cortical column at the molecular level. In all three of these examples, the researchers use high-performance computing alternatives to the cluster-of-commodity-computers distributed architectures that characterize most industrial data warehouses. While the best computing architecture for simulating cortical models may not be clear, it is commonly believed that we either how have or soon will have the computing power to simulate significant portions of cortex at some level of abstraction. This assumes, of course, that we can figure out what the cortex is actually computing.

7 Beyond Early Visual Processing

The grid of columnar processing units which constitutes the primate cortex and the retinotopic maps that characterize the areas participating in early vision, might suggest more familiar engineered vision systems consisting of frame buffers and graphics processors. But this analogy doesn't even apply to the simplest case in which the human subject is staring at a static image. As pointed out earlier, our eyes make large — up to 90° of visual angle — movements several times a second and tiny adjustments much more often.

A typical saccade of, say, 18° of visual angle takes 60–80ms to complete [Harwood, Mezey, and Harris 1999], a period during which we are essentially blind. During the subsequent 200–500ms interval until the next saccade, the image on the fovea is relatively stable, accounting for small adjustments due to micro saccades. So even a rough model for the simplest sort of human visual processing has to be set against the background of two or three fixations per second, each spanning less than half a second, and separated by short — less than 1/10 of a second — periods of blindness.

During each fixation we have 200–500ms in which to make sense of the events projected on the fovea; simplifying enormously, that's time enough to view around 10–15 frames of a video shown at 30 frames per second. In most of our experience, during such a period there is a lot going on in our visual field; our eyes, head and body are often moving and the many objects in our field of view are also in movement, more often than not, moving independent of one another. Either by focusing on a small patch of an object that is motionless relative to our frame of reference or by performing smooth pursuit, we have a brief period in which to analyze what amounts to a very short movie as seen through a tiny aperture. Most individual neurons have receptive fields that span an even smaller spatial and temporal extent.

If we try to interpret movement with too restrictive a spatial extent, we can mistake the direction of travel of a small patch of texture. If we try to work on too restrictive a temporal extent, then we are inundated with small movements many of which are due to noise or uninteresting as they arise from the analog of smooth camera motion. During that half second or so we need to identify stable artifacts, consisting of the orientation, direction, velocity, etc., of small patches of texture and color, and then combine these artifacts to capture features of the somewhat larger region of the fovea we are fixating on. Such a combination need not entail recognizing shape; it could, for example, consist of identifying a set of candidate patches, that may or may not belong to the same object, and summarizing the processing performed during the fixation interval as a collection of statistics pertaining to such patches, including their relative — but not absolute — positions, velocities, etc.

In parallel with processing foveal stimuli, attentional machinery in several neural circuits and, in particular, the lateral intraparietal cortex — which is retinotopically mapped when the eyes are fixated — estimates the saliency of spatial locations throughout the retina, including its periphery where acuity and color sensitivity are poor. These estimates of "interestingness" are used to decide what location to saccade to next. The oculomotor system keeps track of the dislocations associated with each saccade, and this locational information can be fused together using statistics collected over a series of saccades. How such information is combined and the exact nature of the resulting internal representations is largely a mystery.

The main point of the above discussion is that, while human visual processing may begin early in the dorsal and ventral pathways with something vaguely related

to computer image processing using a fixed, spatially-mapped grid of processing and memory units, it very quickly evolves into a process that requires us to combine disjoint intervals of relatively stable imagery into a pastiche from which we can infer properties critical to our survival. Imagine starting with a collection of snapshots taken through a telephoto lens rather than a single high-resolution image taken with a wide-angle lens. This is similar to what several popular web sites do with millions of random, uncalibrated tourist photos.

The neural substrate responsible for performing these combinations must be able to handle a wide range of temporal and spatial scales, numbers and arrangements of inferred parts and surfaces, and a myriad of possible distractions and clutter irrelevant to the task at hand. We know that this processing can be carried out on a more-or-less regular grid of processors — the arrangement of cortical columns is highly suggestive of such a grid. We are even starting to learn the major pathways — bundles of axons sheathed with myelin insulation to speed transmission — connecting these biological processors using diffusion-tensor-imaging techniques. What we don't know is how the cortex allocates its computational resources beyond those areas most directly tied to the peripheral nervous system and that are registered spatially with the locations of the sensors arrayed on the periphery.

From a purely theoretical standpoint, we can simulate any Turing machine with a large enough Boolean circuit, and we can approximate any first-order predicate logic representation that has a finite domain using a propositional representation. Even so, it seems unlikely that even the cortex, with its 10^{11} neurons and 10^{15} connections, has enough capacity to cover the combinatorially many possible arrangements of primitive features that are likely inferred in early vision. This implies that different portions of the cortex must be allocated dynamically to perform processing on very different arrangements of such features.

Bruno Olshausen [1993] theorized that neural circuits could be used to *route* information so that stimuli corresponding to objects and their parts could be transformed to a standard scale and pose, thereby simplifying pattern recognition. Such transformations could, in principle, be carried out by a graphical model. The neural circuitry that serves as the target of such transformations — think of it as a specialized frame buffer of sorts — could be allocated so that different regions are assigned to different parts — this allocation being an instance of the so-called symbol binding problem in connectionist models [Rumelhart and McClelland 1986] of distributed processing.

8 Escaping Retinotopic Tyranny

While much of the computational neuroscience of primate vision seems mired in the first 200 milliseconds or so of early vision when the stimulus is reasonably stable and the image registered on the fovea is mapped retinotopically to areas in V1 through V4, other research on the brain is revealing how we keep track of spatial relationships involving the frames of reference of our head, body, nearby objects, and the larger

world in which we operate. The brain maintains detailed maps of the body and its surrounding physical space in the hippocampus and somatosensory, motor, and parietal cortex [Rizzolatti, Sinigaglia, and Anderson 2007; Blakeslee and Blakeslee 2007]. Recall that the dorsal — "where" and "how" — visual pathway leads to the parietal cortex, which plays an important role in visual attention and our perception of shape. These maps are dynamic, constantly adapting to changes in the body as well as reflecting both short- and long-term knowledge of our surroundings and related spatial relationships.

When attempting to gain insight from biology in building engineered vision systems, it is worth keeping in mind the basic *tasks* of evolved biological vision systems. Much of primate vision serves three broad and overlapping categories of tasks: recognition, navigation and manipulation. Recognition for foraging, mating, and a host of related social and survival tasks; navigation for exploration, localization and controlling territory; manipulation for grasping, climbing, throwing, tool making, etc.

The view [Lengyel 1998] that computer vision is really just inverse graphics ignores the fact that most of these tasks don't require you to be able to construct an accurate 3-D representation of your visual experience. For many recognition tasks it suffices to identify objects, faces, and landmarks you've seen before and associate with these items task-related knowledge gained from prior experience. Navigation to avoid obstacles requires the ability to determine some depth information but not necessarily to recover full 3-D structure. Manipulation is probably the most demanding task in terms of the richness of shape information apparently required, but even so it may be that we are over-emphasizing the role of static shape memory and under-emphasizing the role of dynamic visual servoing — see the discussion in [Rizzolatti, Sinigaglia, and Anderson 2007] for an excellent introduction to what is known about how we understand shape in terms of affordances for manipulation.

But when it comes right down to it, we don't know a great deal about how the visual system handles shape [Tarr and Bülthoff 1998] despite some tantalizing glimpses into what might be going on the inferotemporal cortex [Tsunoda, Yamane, Nishizaki, and Tanifuji 2001; Yamane, Tsunoda, Matsumoto, Phillips, and Tanifuji 2006]. Let's suppose for the sake of discussion that we can build a graphical model of the cortex that handles much of the low-level feature extraction managed by the early visual pathways (V1 through V4) using existing algorithms for performing inference on Markov and conditional random fields and related graphical models. How might we construct a graphical model that captures the part of visual memory that pools together all these low-level features to provide us with such a rich visual experience? Lacking any clear direction from computational neuroscience, we'll take a somewhat unorthodox path from here on out.

As mentioned earlier, several popular web sites offer rich visual experiences that are constructed by combining large image corpora. Photo-sharing web sites like Flickr, Google Picasa and Microsoft Live Labs PhotoSynth are able to combine

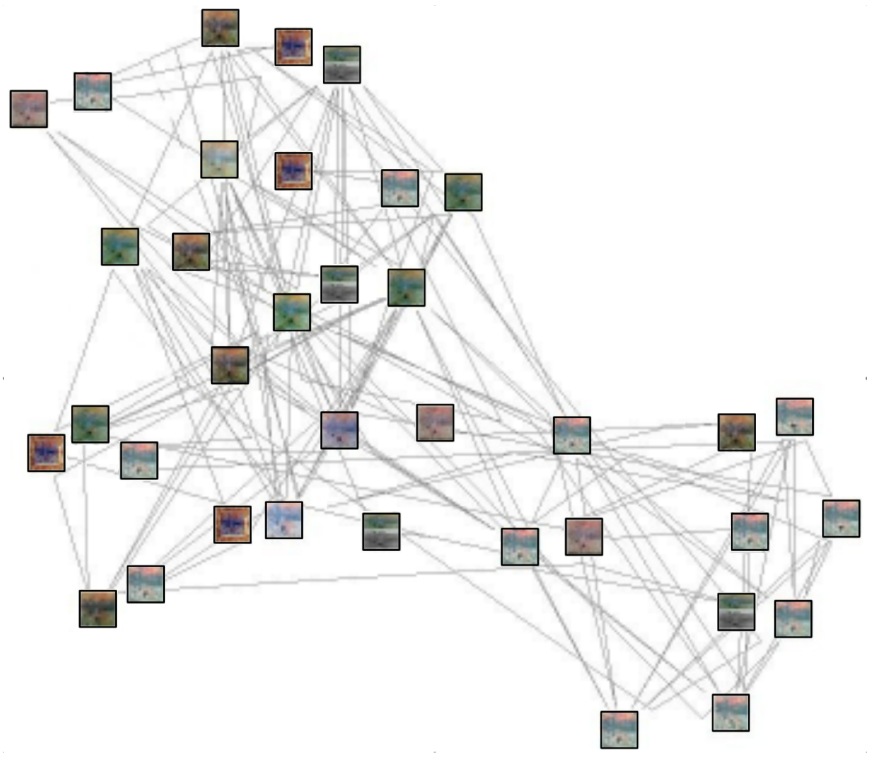


Figure 6. Graphical model with vertices corresponding to image patches and edges representing various relationships among the image patches (after Jing and Baluja [2008]).

multiple snapshots to construct views of popular landmarks from viewpoints not represented by any one snapshot. Google StreetView stitches together video and high-resolution wide-angle images to provide a seamless experience of virtually driving down a street in your home town. Google Earth combines images from satellite, aircraft, ground-based vehicles and, now, sonar-equipped ships and submersibles to allow us explore extensive regions of the planet. These applications are possible due to fast structure-from-motion algorithms that use reliably recoverable and locally distinctive features called *keypoints* extracted from pairs of images to align the images and stitch them together, blending the shared portions and adjusting the color and contrast of the composite to create the illusion of a single image. It is worth noting that these popular web sites facilitate the basic tasks of biological vision that were listed earlier: find me images of a particular famous face, show me a photo of Mount Rainier taken from a site in Tacoma, Washington, tell me what my hotel in New York would look like if I approached it from the direction of Penn Station.

What if the cortex simply memorizes every *novel* fixated foveal patch that spans

some fixed-width receptive field and relates them by using low-level features extracted in V1 through V4 as keypoints to estimate geometric and other meaningful relationships among patches? The use of the word "novel" in this context is meant to convey that some method for statistical pooling of similar patches is required to avoid literally storing every possible patch. This is essentially what Jing and Baluja [2008] do by taking a large corpus of images, extracting low-level features from each image, and then quantifying the similarity between pairs of images by analyzing the features that they have in common. The result is a large graph whose vertices are images and whose edges quantify pair-wise similarity (see Figure 6). By using the low-level features as indices, Jing and Baluja only have to search a small subset of the possible pairs of images, and of those only the ones that pass a specified threshold for similarity are connected by edges. Jing and Baluja further enhance the graph by using a form of spectral graph analysis to rank images in much the same way as Google ranks web pages. Torralba *et al* [2007] have demonstrated that even small image patches contain a great deal of useful information, and furthermore that very large collections of images can be quickly and efficiently searched to retrieve semantically similar images given a target image as a query [Torralba, Fergus, and Weiss 2008].

In principle, such a graph could be represented as a probabilistic graphical model and the spectral analysis reformulated in terms of inference on graphical models. The process whereby the graph is grown over time, incorporating new images and new relationships, currently cannot be formulated as inference on a graphical model, but it is interesting to speculate about very large, yet finite graphs that could evolve over time in response to new evidence. Learning the densities used to quantify the edges in graphical models *can* can be formulated in terms of hyper-parameters directly incorporated into the model and carried out by traditional inference algorithms [Buntine 1994; Heckerman 1995]. Learning graphs whose size and topology change over time is somewhat more challenging to cast in terms of traditional methods for learning graphical models. Graph size is probably not the determining technical barrier however. Very large graphical models consisting of documents, queries, genes, and other entities are now quite common, and, while exact inference in such graphs is typically infeasible, approximate inference is often good enough to provide the foundation for industrial-strength tools.

Unfortunately, there is no way to tie up the many loose ends which have been left dangling in this short survey. Progress depends in part on our better understanding the brain and in particular the parts of the brain that are further from the periphery of the body where our senses are directly exposed to external stimuli. Neuroscience has made significant progress in understanding the brain at the cellular and molecular level, even to the point that we are now able to run large-scale simulations with some confidence that our models reflect important properties of the biology. Computational neuroscientists have also made considerable progress developing models — and graphical models in particular — that account for fea-

tures that appear to play an important role in early visual processing. The barrier to further progress seems to be the same impediment that we run into in so many other areas of computer vision, machine learning and artificial intelligence more generally, namely the problem of representation. How and what does the brain represent about the blooming, buzzing world in which we are embedded? The answer to that question will take some time to figure out, but no doubt probabilistic graphical models will continue to provide a powerful tool in this inquiry, thanks in no small measure to the work of Judea Pearl, his students and his many collaborators.

References

Anderson, J. and J. Sutton (1997). If we compute faster, do we understand better? *Behavior Ressearch Methods, Instruments and Computers 29*, 67–77.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, pp. 153–160. Cambridge, MA: MIT Press.

Bienenstock, E. and S. Geman (1995). Compositionality in neural systems. In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 223–226. Bradford Books/MIT Press.

Blakeslee, S. and M. Blakeslee (2007). *The Body Has a Mind of Its Own*. Random House.

Brady, N. and D. J. Field (2000). Local contrast in natural images: normalisation and coding efficiency. *Perception 29*(9), 1041–1055.

Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research 2*, 159–225.

Cadieu, C. and B. Olshausen (2008). Learning transformational invariants from time-varying natural images. In D. Schuurmans and Y. Bengio (Eds.), *Advances in Neural Information Processing Systems 21*. Cambridge, MA: MIT Press.

Dean, T. (2005). A computational model of the cerebral cortex. In *Proceedings of AAAI-05*, Cambridge, Massachusetts, pp. 938–943. MIT Press.

Dean, T. (2006, August). Learning invariant features using inertial priors. *Annals of Mathematics and Artificial Intelligence 47*(3-4), 223–250.

Dean, T., G. Corrado, and R. Washington (2009, December). Recursive sparse, spatiotemporal coding. In *Proceedings of the Fifth IEEE International Workshop on Multimedia Information Processing and Retrieval*.

Dean, T. and M. Wellman (1991). *Planning and Control*. San Francisco, California: Morgan Kaufmann Publishers.

DeGroot, M. (1970). *Optimal Statistical Decisions*. New York: McGraw-Hill.

DeGroot, M. H. (1986). *Probability and Statistics*. Reading, MA: Second edition, Addison-Wesley.

Dehaene, S. (2009). *Reading in the Brain: The Science and Evolution of a Human Invention*. Viking Press.

Fine, S., Y. Singer, and N. Tishby (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning 32*(1), 41–62.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation 3*, 194–200.

George, D. and J. Hawkins (2005). A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the International Joint Conference on Neural Networks*, Volume 3, pp. 1812–1817. IEEE.

Harwood, M. R., L. E. Mezey, and C. M. Harris (1999). The spectral main sequence of human saccades. *The Journal of Neuroscience 19*, 9098–9106.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. Technical Report MSR-95-06, Microsoft Research.

Hinton, G. and R. Salakhutdinov (2006, July). Reducing the dimensionality of data with neural networks. *Science 313*(5786), 504–507.

Hinton, G. E. (2005). What kind of a graphical model is the brain? In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*.

Hoffman, D. (1998). *Visual Intelligence: How We Create What we See*. New York, NY: W. W. Norton.

Hoiem, D., A. Efros, and M. Hebert (2007). Recovering surface layout from an image. *International Journal of Computer Vision 75*(1), 151–172.

Hubel, D. H. and T. N. Wiesel (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology 160*, 106–154.

Hubel, D. H. and T. N. Wiesel (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology 195*, 215–243.

Hyvärinen, A., J. Hurri, and J. Väyrynen (2003). Bubbles: a unifying framework for low-level statistical properties of natural image sequences. *Journal of the Optical Society of America 20*(7), 1237–1252.

Izhikevich, E. M. and G. M. Edelman (2008). Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Science 105*(9), 3593–3598.

Jin, Y. and S. Geman (2006). Context and hierarchy in a probabilistic image model. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, Volume 2, pp. 2145–2152. IEEE Computer Society.

Jing, Y. and S. Baluja (2008). Pagerank for product image search. In *Proceedings of the 17th World Wide Web Conference*.

Kivinen, J. J., E. B. Sudderth, and M. I. Jordan (2007). Learning multiscale representations of natural scenes using dirichlet processes. In *Proceedings of the 11th IEEE International Conference on Computer Vision*.

Konen, C. S. and S. Kastner (2008). Two hierarchically organized neural systems for object information in human visual cortex. *Nature Neuroscience 11*(2), 224–231.

LeCun, Y. and Y. Bengio (1995). Convolutional networks for images, speech, and time-series. In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*. Bradford Books/MIT Press.

Lee, T. S. and D. Mumford (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America 2*(7), 1434–1448.

Lengyel, J. (1998). The convergence of graphics and vision. *Computer 31*(7), 46–53.

Mountcastle, V. B. (2003, January). Introduction to the special issue on computation in cortical columns. *Cerebral Cortex 13*(1), 2–4.

Mumford, D. (1991). On the computational architecture of the neocortex I: The role of the thalamo-cortical loop. *Biological Cybernetics 65*, 135–145.

Mumford, D. (1992). On the computational architecture of the neocortex II: The role of cortico-cortical loops. *Biological Cybernetics 66*, 241–251.

Newman, M., D. Watts, and S. Strogatz (2002). Random graph models of social networks. *Proceedings of the National Academy of Science 99*, 2566–2572.

Olshausen, B. and C. Cadieu (2007). Learning invariant and variant components of time-varying natural images. *Journal of Vision 7*(9), 964–964.

Olshausen, B. A., A. Anderson, and D. C. Van Essen (1993). A neurobiological model of visual attention and pattern recognition based on dynamic routing of information. *Journal of Neuroscience 13*(11), 4700–4719.

Olshausen, B. A. and D. J. Field (2005). How close are we to understanding V1? *Neural Computation 17*, 1665–1699.

Osindero, S. and G. Hinton (2008). Modeling image patches with a directed hierarchy of markov random fields. In J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, pp. 1121–1128. Cambridge, MA: MIT Press.

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

Ranzato, M., Y. Boureau, and Y. LeCun (2007). Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Riesenhuber, M. and T. Poggio (1999, November). Hierarchical models of object recognition in cortex. *Nature Neuroscience 2*(11), 1019–1025.

Rizzolatti, G., C. Sinigaglia, and F. Anderson (2007). *Mirrors in the Brain How Our Minds Share Actions, Emotions, and Experience*. Oxford, UK: Oxford University Press.

Rumelhart, D. E. and J. L. McClelland (Eds.) (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume I: Foundations*. Cambridge, Massachusetts: MIT Press.

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Second edition, Prentice Hall.

Saxena, A., S. Chung, and A. Ng (2007). 3-D depth reconstruction from a single still image. *International Journal of Computer Vision 76*(1), 53–69.

Sporns, O. and J. D. Zwi (2004). The small world of the cerebral cortex. *Neuroinformatics 2*(2), 145–162.

Tarr, M. and H. Bülthoff (1998). Image-based object recognition in man, monkey and machine. *Cognition 67*, 1–20.

Tenenbaum, J. and H. Barrow (1977). Experiments in interpretation-guided segmentation. *Artificial Intelligence 8*, 241–277.

Torralba, A., R. Fergus, and W. Freeman (2007). Object and scene recognition in tiny images. *Journal of Vision 7*(9), 193–193.

Torralba, A., R. Fergus, and Y. Weiss (2008). Small codes and large image databases for recognition. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 1–8. IEEE Computer Society.

Tsunoda, K., Y. Yamane, M. Nishizaki, and M. Tanifuji (2001). Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. *Nature Neuroscience 4*, 832–838.

Ullman, S. and S. Soloviev (1999). Computation of pattern invariance in brain-like structures. *Neural Networks 12*, 1021–1036.

Ullman, S., M. Vidal-Naquet, and E. Sali (2002). Visual features of intermediate complexity and their use in classification. *Nature Neuroscience 5*(7), 682–687.

Wiskott, L. and T. Sejnowski (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation 14*(4), 715–770.

Yamane, Y., K. Tsunoda, M. Matsumoto, N. A. Phillips, and M. Tanifuji (2006). Representation of the spatial relationship among object parts by neurons in macaque inferotemporal cortex. *Journal Neurophysiology 96*, 3147–3156.

9

On the Power of Belief Propagation: A Constraint Propagation Perspective

R. Dechter, B. Bidyuk, R. Mateescu and E. Rollon

1 Introduction

In his seminal paper, Pearl [1986] introduced the notion of Bayesian networks and the first processing algorithm, *Belief Propagation (BP)*, that computes posterior marginals, called beliefs, for each variable when the network is singly connected. The paper provided the foundation for the whole area of Bayesian networks. It was the first in a series of influential papers by Pearl, his students and his collaborators that culminated a few years later in his book on probabilistic reasoning [Pearl 1988]. In his early paper Pearl showed that for singly connected networks (e.g., polytrees) the distributed message-passing algorithm converges to the correct marginals in a number of iterations equal to the diameter of the network. In his book Pearl goes further to suggest the use of BP for loopy networks as an approximation algorithm (see page 195 and exercise 4.7 in [Pearl 1988]). During the decade that followed researchers focused on extending BP to general loopy networks using two principles. The first is tree-clustering, namely, the transformation of a general network into a tree of large-domain variables called clusters on which BP can be applied. This led to the join-tree or junction-tree clustering and to the bucket-elimination schemes [Pearl 1988; Dechter 2003] whose time and space complexity is exponential in the tree-width of the network. The second principle is that of cutset-conditioning that decomposes the original network into a collection of independent singly-connected networks all of which must be processed by BP. The cutset-conditioning approach is time exponential in the network's loop-cutset size and require linear space [Pearl 1988; Dechter 2003].

The idea of applying belief propagation directly to multiply connected networks caught up only a decade after the book was published, when it was observed by researchers in coding theory that high performing probabilistic decoding algorithms such as turbo codes and low density parity-check codes, which significantly outperformed the best decoders at the time, are equivalent to an iterative application of Pearl's belief propagation algorithm [McEliece, MacKay, and Cheng 1998]. This success intrigued researchers and started massive explorations of the potential of these local computation algorithms for general applications. There is now a significant body of research seeking the understanding and improvement of the inference power of iterative belief propagation (IBP).

The early work on IBP showed its convergence for a single loop, provided empirical evidence of its successes and failures on various classes of networks [Rish, Kask, and Dechter 1998; Murphy, Weiss, and Jordan 2000] and explored the relationship between energy minimization and belief-propagation shedding light on convergence and stable points [Yedidia, Freeman, and Weiss 2000]. Current state of the art in convergence analysis are the works by [Ihler, Fisher, and Willsky 2005; Mooij and Kappen 2007] that characterize convergence in networks having no determinism. The work by [Roosta, Wainwright, and Sastry 2008] also includes an analysis of the possible effects of strong evidence on convergence which can act to suppress the effects of cycles. As far as accuracy, the work of [Ihler 2007] considers how weak potentials can make the graph sufficiently tree-like to provide error bounds, a work which is extended and improved in [Mooij and Kappen 2009]. For additional information see [Koller 2010].

While a significant progress has been made in understanding the relationship between belief propagation and energy minimization, and while many extensions and variations were proposed, some with remarkable performance (e.g., survey propagation for solving satisfiability for random SAT problems), the following questions remain even now:

- Why does belief propagation work so well on coding networks?
- Can we characterize additional classes of problems for which IBP is effective?
- Can we assess the quality of the algorithm's performance once and if it converges.

In this paper we try to shed light on the power (and limits) of belief propagation algorithms and on the above questions by explicating its relationship with constraint propagation algorithms such as arc-consistency. Our results are relevant primarily to networks that have determinism and extreme probabilities. Specifically, we show that: (1) Belief propagation converges for zero beliefs; (2) All IBP-inferred zero beliefs are correct; (3) IBP's power to infer zero beliefs is as weak and as strong as that of arc-consistency; (4) Evidence and inferred singleton beliefs act like cutsets during IBP's performance. From points (2) and (4) it follows that if the inferred evidence breaks all the cycles, then IBP converges to the exact beliefs for all variables.

Subsequently, we investigate empirically the behavior of IBP for inferred near-zero beliefs. Specifically, we explore the hypothesis that: (5) If IBP infers that the belief of a variable is close to zero then this inference is relatively accurate. We will see that while our empirical results support the hypothesis on benchmarks having no determinism, the results are quite mixed for networks with determinism.

Finally, (6) We investigate if variables that have extreme probabilities in all its domain values (i.e., extreme support) also nearly cut off information flow. If that hypothesis is true, whenever the set of variables with extreme support constitute a

loop-cutset, IBP is likely to converge and, if the inferred beliefs for those variables are sound, it will converge to accurate beliefs throughout the network.

On coding networks that posses significant determinism, we do see this desired behavior. So, we could view this hypothesis as the first to provide a plausible explanation to the success of belief propagation on coding networks. In coding networks the channel noise is modeled through a normal distribution centered at the transmitted character and controlled by a small standard deviation. The problem is modeled as a layered belief network whose sink nodes are all evidence that transmit extreme support to their parents, which constitute all the rest of the variables. The remaining dependencies are functional and arc-consistency on this type of networks is strong and often complete. Alas, as we show, on some other deterministic networks IBP's performance inferring near zero values is utterly inaccurate, and therefore the strength of this explanation is questionable.

The paper is based for the most part on [Dechter and Mateescu 2003] and also on [Bidyuk and Dechter 2001]. The empirical portion of the paper includes significant new analysis of recent empirical evaluations carried on in UAI 2006 and UAI 2008[1].

2 Arc-consistency

DEFINITION 1 (constraint network). A constraint network $\mathcal{C}$ is a triple $\mathcal{C} = \langle X, D, C \rangle$, where $X = \{X_1, ..., X_n\}$ is a set of variables associated with a set of discrete-valued domains $D = \{D_1, ..., D_n\}$ and a set of constraints $C = \{C_1, ..., C_r\}$. Each constraint C_i is a pair $\langle S_i, R_i \rangle$ where R_i is a relation $R_i \subseteq D_{S_i}$ defined on a subset of variables $S_i \subseteq X$ and D_{S_i} is the Cartesian product of the domains of variables S_i. The relation R_i denotes all tuples of D_{S_i} allowed by the constraint. The projection operator π creates a new relation, $\pi_{S_j}(R_i) = \{x \mid x \in D_{S_j}$ and $\exists y, y \in D_{S_i \setminus S_j}$ and $x \cup y \in R_i\}$, where $S_j \subseteq S_i$. Constraints can be combined with the join operator $\bowtie$, resulting in a new relation, $R_i \bowtie R_j = \{x \mid x \in D_{S_i \cup S_j}$ and $\pi_{S_i}(x) \in R_i$ and $\pi_{S_j}(x) \in R_j\}$.

DEFINITION 2 (constraint satisfaction problem). The constraint satisfaction problem (CSP) defined over a constraint network $\mathcal{C} = \langle X, D, C \rangle$, is the task of finding a solution, that is, an assignment of values to all the variables $x = (x_1, ..., x_n), x_i \in D_i$, such that $\forall C_i \in C, \pi_{S_i}(x) \in R_i$. The set of all solutions of the constraint network $\mathcal{C}$ is $sol(\mathcal{C}) = \bowtie R_i$.

2.1 Describing Arc-Consistency Algorithms

Arc-consistency algorithms belong to the well-known class of constraint propagation algorithms [Mackworth 1977; Dechter 2003]. All constraint propagation algorithms are polynomial time algorithms that are at the center of constraint processing techniques.

DEFINITION 3 (arc-consistency). [Mackworth 1977] Given a binary constraint net-

[1] http://graphmod.ics.uci.edu/uai08/Evaluation/Report

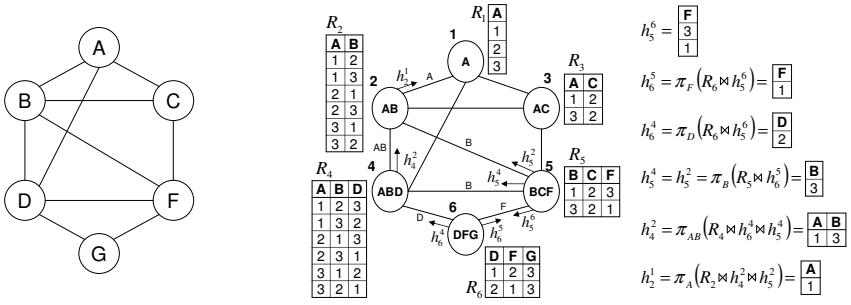


Figure 1. Part of the execution of RDAC algorithm

work $\mathcal{C} = \langle X, D, C \rangle$, $\mathcal{C}$ is arc-consistent iff for every binary constraint $R_i \in C$ s.t. $S_i = \{X_j, X_k\}$, every value $x_j \in D_j$ has a value $x_k \in D_k$ s.t. $(x_j, x_k) \in R_i$.

When a binary constraint network is not arc-consistent, arc-consistency algorithms remove values from the domains of the variables until an arc-consistent network is generated. A variety of such algorithms were developed over the past three decades [Dechter 2003]. We will consider here a simple and not the most efficient version, which we call *relational distributed arc-consistency* algorithm. Rather than defining it on binary constraint networks we will define it directly over the dual graph, extending the arc-consistency condition to non-binary networks.

DEFINITION 4 (dual graph). Given a set of functions/constraints $F = \{f_1, ..., f_r\}$ over scopes $S_1, ..., S_r$, the dual graph of F is a graph $\mathcal{D}_F = (V, E, L)$ that associates a node with each function, namely $V = F$, and an arc connects any two nodes whose scope share a variable, $E = \{(f_i, f_j) | S_i \cap S_j \neq \emptyset\}$. L is a set of labels for the arcs, where each arc is labeled by the shared variables of its nodes, $L = \{l_{ij} = S_i \cap S_j | (i, j) \in E\}$.

Algorithm *Relational distributed arc-consistency* (RDAC) is a message passing algorithm defined over the dual graph $\mathcal{D}_C$ of a constraint network $\mathcal{C} = \langle X, D, C \rangle$. It enforces what is known as *relational arc-consistency* [Dechter 2003]. Each node (a constraint) in $\mathcal{D}_{C_i}$, for a constraint $C_i \in \mathcal{C}$ maintains a current set of viable tuples R_i. Let $ne(i)$ be the set of neighbors of C_i in $\mathcal{D}_C$. Every node C_i sends a message to any node $C_j \in ne(i)$, which consists of the tuples over their label variables l_{ij} that are allowed by the current relation R_i. Formally, let R_i and R_j be two constraints sharing scopes, whose arc in $\mathcal{D}_C$ is labeled by l_{ij}. The message that R_i sends to R_j denoted h_i^j is defined by:

(1) $h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i))$

and each node updates its current relation according to:

(2) $R_i \leftarrow R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)$

EXAMPLE 5. Figure 1 describes part of the execution of RDAC for a graph col-

oring problem, having the constraint graph shown on the left. All variables have the same domain, {1,2,3}, except for variable C whose domain is 2, and variable G whose domain is 3. The arcs correspond to *not equal* constraints, and the relations are R_A, R_{AB}, R_{AC}, R_{ABD}, R_{BCF}, R_{DFG}, where the subscript corresponds to their scopes. The dual graph of this problem is given on the right side of the figure, and each table shows the initial constraints (there are unary, binary and ternary constraints). To initialize the algorithm, the first messages sent out by each node are universal relations over the labels. For this example, RDAC actually solves the problem and finds the unique solution A=1, B=3, C=2, D=2, F=1, G=3.

Relational distributed arc-consistency algorithm converges after $O(r \cdot t)$ iterations to the largest relational arc-consistent network that is equivalent to the original network, where r is the number of constraints and t bounds the number of tuples in each constraint. Its complexity can be shown to be $O(r^2 t^2 \log t)$ [Dechter 2003].

3 Iterative Belief Propagation

DEFINITION 6 (belief network). A belief network is a quadruple $\mathcal{B} = \langle X, D, G, P \rangle$ where $X = \{X_1, \ldots, X_n\}$ is a set of random variables, $D = \{D_1, ..., D_n\}$ is the set of the corresponding domains, $G = (X, E)$ is a directed acyclic graph over X and $P = \{p_1, ..., p_n\}$ is a set of conditional probability tables (CPTs) $p_i = P(X_i|pa(X_i))$, where $pa(X_i)$ are the parents of X_i in G. The belief network represents a probability distribution over X having the product form $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_{pa(X_i)})$. An evidence set e is an instantiated subset of variables. The family of X_i, denoted by $fa(X_i)$, includes X_i and its parent variables. Namely, $fa(X_i) = \{X_i\} \cup pa(X_i)$.

DEFINITION 7 (belief updating problem). The belief updating problem defined over a belief network $\mathcal{B} = \langle X, D, G, P \rangle$ is the task of computing the posterior probability $P(Y|e)$ of *query* nodes $Y \subseteq X$ given evidence e. We will sometime denote by P_B the exact probability according the Baysian network B. When Y consists of a single variable X_i, $P_B(X_i|e)$ is also denoted as $Bel(X_i)$ and called belief, or posterior marginal, or just marginal.

3.1 Describing Iterative Belief Propagation

Iterative belief propagation (IBP) is an iterative application of Pearl's algorithm that was defined for poly-trees [Pearl 1988]. Since it is a distributed algorithm, it is well defined for any network. We will define IBP as operating over the belief network's dual join-graph.

DEFINITION 8 (dual join-graph). Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, a dual join-graph is an arc subgraph of the dual graph $\mathcal{D}_B$ whose arc labels are subsets of the labels of $\mathcal{D}_B$ satisfying the *running intersection property*, namely, that any two nodes that share a variable in the dual join-graph be connected by a path of arcs whose labels *contain* the shared variable. An *arc-minimal* dual join-graph is one for which none of the labels can be further reduced while maintaining the running

Algorithm **IBP**
Input: An arc-labeled dual join-graph $DJ = (V, E, L)$ for a belief network $\mathcal{B} = \langle X, D, G, P \rangle$. Evidence e.
Output: An augmented graph whose nodes include the original CPTs and the messages received from neighbors. Approximations of $P(X_i|e)$ and $P(fa(X_i)|e)$, $\forall X_i \in X$.
Denote by: h_u^v the message from u to v; $ne(u)$ the neighbors of u in V; $ne_v(u) = ne(u) - \{v\}$; l_{uv} the label of $(u, v) \in E$; $elim(u, v) = fa(X_i) - fa(X_j)$, where u and v are the vertexs of family $fa(X_i)$ and $fa(X_j)$ in DJ, respectively.
• **One iteration of IBP**
For every node u in DJ in a topological order and back, do:
1. **Process observed variables**
Assign evidence variables to each p_i and remove them from the labeled arcs.
2. **Compute and send to v the function:**

$$h_u^v = \sum_{elim(u,v)} (p_u \cdot \prod_{\{h_i^u, i \in ne_v(u)\}} h_i^u)$$

EndFor
• **Compute approximations of $P(X_i|e)$ and $P(fa(X_i)|e)$:**
For every $X_i \in X$ (let u be the vertex of family $fa(X_i)$ in DJ), do:
$P(fa(X_i)|e) = \alpha(\prod_{h_i^u, u \in ne(i)} h_i^u) \cdot p_u$
$P(X_i|e) = \alpha \sum_{fa(X_i) - \{X_i\}} P(fa(X_i)|e)$
EndFor

Figure 2. Algorithm Iterative Belief Propagation

intersection property.

In IBP each node in the dual join-graph sends a message over an adjacent arc whose scope is identical to its label. Pearl's original algorithm sends messages whose scopes are singleton variables only. It is easy to show that any dual graph (which itself is a dual join-graph) has an arc-minimal singleton dual join-graph which can be constructed directly by labeling the arc between the CPT of a variable and the CPT of its parent, by its parent variable. Algorithm IBP defined for any dual join-graph is given in Figure 2. One iteration of IBP is time and space linear in the size of the belief network, and when IBP is applied to the singleton labeled dual graph it coincides with Pearl's belief propagation. The inferred approximation of belief $P(X|e)$ output by IBP, will be denoted by $P_{IBP}(X|e)$.

4 Belief Propagation's Inferred Zeros

We will now make connections between distributed relational arc-consistency and iterative belief propagation. We first associate any belief network with a constraint network that captures its zero probability tuples and define algorithm IBP-RDAC, an IBP-like algorithm that achieves relational arc-consistency on the associated constraint network. Then, we show that IBP-RDAC and IBP are equivalent in terms of removing inconsistent domain values and computing zero marginal probabilities, respectively. Since arc-consistency algorithms are well understood, this correspon-

dence between IBP-RDAC and IBP yields the main claims and provides insight into the behavior of IBP for inferred zero beliefs. In particular, this relationship justifies the iterative application of belief propagation algorithms, while also illuminates their "distance" from being complete.

More precisely, in this section we will show that: (a) If a variable-value pair is assessed in some iteration by IBP as having a zero-belief, it remains zero in subsequent iterations; (b) Any IBP-inferred zero-belief is correct with respect to the corresponding belief network's marginal; and (c) IBP converges in finite time for all its inferred zeros.

4.1 Flattening the Belief Network

Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, we define the flattening of a belief network $\mathcal{B}$, called $flat(\mathcal{B})$, as the constraint network where all the zero entries in a probability table are removed from the corresponding relation. Formally,

DEFINITION 9 (flattening). Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, its flattening is a constraint network $flat(\mathcal{B}) = \langle X, D, flat(P) \rangle$. Each CPT $p_i \in P$ over $fa(X_i)$ is associated with a constraint $\langle S_i, R_i \rangle$ s.t. $S_i = fa(X_i)$ and $R_i = \{(x_i, x_{pa(X_i)}) \in D_{S_i} | P(x_i|x_{pa(X_i)}) > 0\}$. The set $flat(P)$ is the set of the constraints $\langle S_i, R_i \rangle$, $\forall p_i \in P$.

EXAMPLE 10. Figure 3 shows (a) a belief network and (b) its corresponding flattening.

THEOREM 11. *Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, where $X = \{X_1, \ldots, X_n\}$, for any tuple $x = (x_1, \ldots, x_n)$: $P_{\mathcal{B}}(x) > 0 \Leftrightarrow x \in sol(flat(\mathcal{B}))$, where $sol(flat(\mathcal{B}))$ is the set of solutions of $flat(B)$.*

Proof. $P_{\mathcal{B}}(x) > 0 \Leftrightarrow \Pi_{i=1}^{n} P(x_i|x_{pa(X_i)}) > 0 \Leftrightarrow \forall i \in \{1, \ldots, n\},\ P(x_i|x_{pa(X_i)}) > 0$ $\Leftrightarrow \forall i \in \{1, \ldots, n\},\ (x_i, x_{pa(X_i)}) \in R_{F_i} \Leftrightarrow x \in sol(flat(\mathcal{B}))$. □

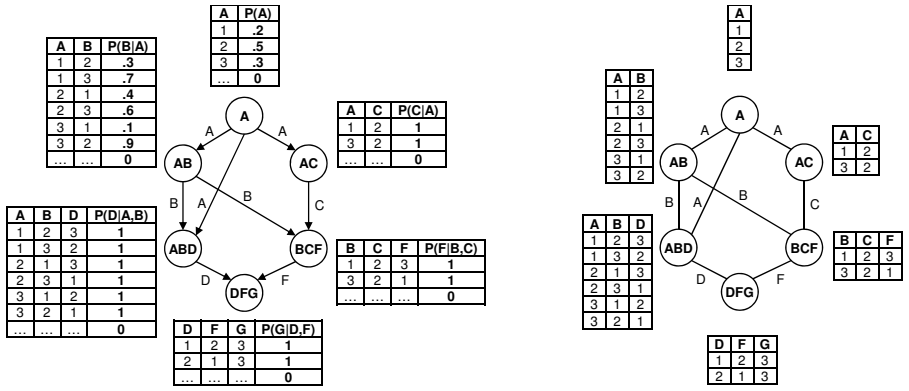
Clearly this can extend to Bayesian networks with evidence:

COROLLARY 12. *Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, and evidence e $P_B(x|e) > 0 \Leftrightarrow x \in sol(flat(B) \wedge e)$.*

We next define algorithm IBP-RDAC and show that it achieves relational arc-consistency on the flat network.

DEFINITION 13 (Algorithm IBP-RDAC). Given $\mathcal{B} = \langle X, D, G, P \rangle$ and evidence e, let $\mathcal{D}_{\mathcal{B}}$ be a dual join-graph and $\mathcal{D}_{flat(\mathcal{B})}$ be a corresponding dual join-graph of the constraint network $flat(\mathcal{B})$. Algorithm IBP-RDAC applied to $\mathcal{D}_{flat(\mathcal{B})}$ is defined using IBP's specification in Figure 2 with the following modifications:

1. Pre-processing evidence: when processing evidence, we remove from each $R_i \in flat(P)$ those tuples that do not agree with the assignments in evidence e.
2. Instead of $\prod$, we use the join operator $\bowtie$.



(a) Bayesian network (b) Flat constraint network

Figure 3. Flattening of a Bayesian network

3. Instead of $\sum$, we use the projection operator π.

4. At the termination, we update the domains of variables by:

$$D_i \leftarrow D_i \cap \pi_{X_i}((\bowtie_{v \in ne(u)} h_{(v,u)}) \bowtie R_i)$$

By construction, it should be easy to see that,

PROPOSITION 14. *Given a belief network $\mathcal{B} = \langle X, D, G, P \rangle$, algorithm IBP-RDAC is identical to algorithm RDAC when applied to $\mathcal{D}_{flat(\mathcal{B})}$. Therefore, IBP-RDAC enforces relational arc-consistency over $flat(\mathcal{B})$.*

Due to the convergence of RDAC, we get that:

PROPOSITION 15. *Given a belief network $\mathcal{B}$, algorithm IBP-RDAC over $flat(\mathcal{B})$ converges in $O(n \cdot t)$ iterations, where n is the number of nodes in $\mathcal{B}$ and t is the maximum number of tuples over the labeling variables between two nodes that have positive probability.*

4.2 The Main Claim

In the following we will establish an equivalence between IBP and IBP-RDAC in terms of zero probabilities.

PROPOSITION 16. *When IBP and IBP-RDAC are applied in the same order of computation to $\mathcal{B}$ and $flat(\mathcal{B})$ respectively, the messages computed by IBP are identical to those computed by IBP-RDAC in terms of zero / non-zero probabilities. That is, for any pair of corresponding messages, $h_{(u,v)}(t) \neq 0$ in IBP iff $t \in h_{(u,v)}$ in IBP-RDAC.*

Proof. The proof is by induction. The base case is trivially true since messages h in IBP are initialized to a uniform distribution and messages h in IBP-RDAC are initialized to complete relations.

The induction step. Suppose that $h^{IBP}_{(u,v)}$ is the message sent from u to v by IBP. We will show that if $h^{IBP}_{(u,v)}(x) \neq 0$, then $x \in h^{IBP-RDAC}_{(u,v)}$ where $h^{IBP-RDAC}_{(u,v)}$ is the message sent by IBP-RDAC from u to v. Assume that the claim holds for all messages received by u from its neighbors. Let $f \in u$ in IBP and R_f be the corresponding relation in IBP-RDAC, and t be an assignment of values to variables in $elim(u,v)$. We have $h^{IBP}_{(u,v)}(x) \neq 0 \Leftrightarrow \sum_{elim(u,v)} \prod_f f(x) \neq 0 \Leftrightarrow \exists t, \prod_f f(x,t) \neq 0 \Leftrightarrow \exists t, \forall f, f(x,t) \neq 0 \Leftrightarrow \exists t, \forall f, \pi_{scope(R_f)}(x,t) \in R_f \Leftrightarrow \exists t, \pi_{elim(u,v)}(\bowtie_{R_f} \pi_{scope(R_f)}(x,t)) \in h^{IBP-RDAC}_{(u,v)} \Leftrightarrow x \in h^{IBP-RDAC}_{(u,v)}$. □

Moving from tuples to domain values, we will show that whenever IBP computes a marginal probability $P_{IBP}(x_i|e) = 0$, IBP-RDAC removes x_i from the domain of variable X_i, and vice-versa.

PROPOSITION 17. *Given a belief network $\mathcal{B}$ and evidence e, IBP applied to $\mathcal{B}$ derives $P_{IBP}(x_i|e) = 0$ iff IBP-RDAC over $flat(\mathcal{B})$ decides that $x_i \notin D_i$.*

Proof. According to Proposition 16, the messages computed by IBP and IBP-RDAC are identical in terms of zero probabilities. Let $f \in cluster(u)$ in IBP and R_f be the corresponding relation in IBP-RDAC, and t be an assignment of values to variables in $\chi(u) \backslash X_i$. We will show that when IBP computes $P(X_i = x_i) = 0$ (upon convergence), then IBP-RDAC computes $x_i \notin D_i$. We have $P(X_i = x_i) = \sum_{X \backslash X_i} \prod_f f(x_i) = 0 \Leftrightarrow \forall t, \prod_f f(x_i,t) = 0 \Leftrightarrow \forall t, \exists f, f(x_i,t) = 0 \Leftrightarrow$
$\forall t, \exists R_f, \pi_{scope(R_f)}(x_i,t) \notin R_f \Leftrightarrow \forall t, (x_i,t) \notin (\bowtie_{R_f} R_f(x_i,t)) \Leftrightarrow x_i \notin D_i \cap \pi_{X_i}(\bowtie_{R_f} R_f(x_i,t)) \Leftrightarrow x_i \notin D_i$. Since arc-consistency is sound, so is the decision of zero probabilities. □

We can now conclude that:

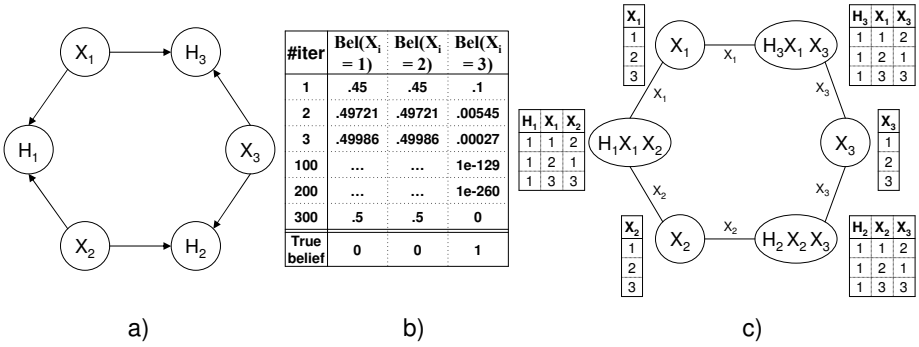
THEOREM 18. *Given evidence e, whenever IBP applied to $\mathcal{B}$ infers $P_{IBP}(x_i|e) = 0$, the marginal $Bel(x_i) = P_{\mathcal{B}}(x_i|e) = 0$.*

Proof. By Proposition 17, if IBP over $\mathcal{B}$ computes $P_{IBP}(x_i|e) = 0$, then IBP-RDAC over $flat(\mathcal{B})$ removes the value x_i from the domain D_i. Therefore, $x_i \in D_i$ is a no-good of the constraint network $flat(\mathcal{B})$ and from Theorem 11 it follows that $Bel(x_i) = 0$. □

Next, we show that the time it takes IBP to find its inferred zeros is bounded.

PROPOSITION 19. *Given a belief network $\mathcal{B}$ and evidence e, IBP finds all its x_i for which $P_{IBP}(x_i|e) = 0$ in finite time, that is, there exists a number k such that no $P_{IBP}(x_i|e) = 0$ will be generated after k iterations.*

Proof. This follows from the fact that the number of iterations it takes for IBP to compute $P_{IBP}(X_i = x_i|e) = 0$ over $\mathcal{B}$ is exactly the same number of iterations IBP-RDAC needs to remove x_i from the domain D_i over $flat(\mathcal{B})$ (Propositions 16 and 17) and the fact that IBP-RDAC's number of iterations is bounded (Proposition 15). □



#iter	Bel(X_i = 1)	Bel(X_i = 2)	Bel(X_i = 3)
1	.45	.45	.1
2	.49721	.49721	.00545
3	.49986	.49986	.00027
100	...	...	1e-129
200	...	...	1e-260
300	.5	.5	0
True belief	0	0	1

Figure 4. a) A belief network; b) Example of a finite precision problem; and (c) An arc-minimal dual join-graph.

4.3 A Finite Precision Problem

Algorithms should always be implemented with care on finite precision machines. In the following example we show that IBP's messages converge in the limit (i.e. in an infinite number of iterations), but they do not stabilize in any finite number of iterations.

EXAMPLE 20. Consider the belief network in Figure 4a defined over 6 variables X_1, X_2, X_3, H_1, H_2, H_3. The domain of the X variables is $\{1, 2, 3\}$ and the domain of the H variables is $\{0, 1\}$. The priors on X variables are:

$$P(X_i) = \begin{cases} 0.45, & if \;\; X_i = 1; \\ 0.45, & if \;\; X_i = 2; \\ 0.1, & if \;\; X_i = 3; \end{cases}$$

There are three CPTs over the scopes: $\{H_1, X_1, X_2\}$, $\{H_2, X_2, X_3\}$, and $\{H_3, X_1, X_3\}$. The values of the CPTs for every triplet of variables $\{H_k, X_i, X_j\}$ are:

$$\begin{aligned} P(h_k = 1|x_i, x_j) &= \begin{cases} 1, & if \;\; (3 \neq x_i \neq x_j \neq 3); \\ 1, & if \;\; (x_i = x_j = 3); \\ 0, & otherwise \;\; ; \end{cases} \\ P(h_k = 0|x_i, x_j) &= 1 - P(h_k = 1|x_i, x_j). \end{aligned}$$

Consider the evidence set $e = \{H_1 = H_2 = H_3 = 1\}$. This Bayesian network expresses the probability distribution that is concentrated in a single tuple:

$$P(x_1, x_2, x_3|e) = \begin{cases} 1, & if \;\; x_1 = x_2 = x_3 = 3; \\ 0, & otherwise. \end{cases}$$

The belief for any of the X variables as a function of the number of iteration is given in Figure 4b. After about 300 iterations, the finite precision of our computer is not able to represent the value for $Bel(X_i = 3)$, and this appears to be zero,

yielding the final updated belief $(.5, .5, 0)$, when in fact the true updated belief should be $(0, 0, 1)$. Notice that $(.5, .5, 0)$ cannot be regarded as a legitimate fixed point for IBP. Namely, if we would initialize IBP with the values $(.5, .5, 0)$, then the algorithm would maintain them, appearing to have a fixed point. However, initializing IBP with zero values cannot be expected to be correct. Indeed, when we initialize with zeros we forcibly introduce determinism in the model, and IBP will always maintain it afterwards.

However, this example does not contradict our theory because, mathematically, $Bel(X_i = 3)$ never becomes a true zero, and IBP never reaches a quiescent state. The example shows however that a close to zero inferred belief by IBP can be arbitrarily inaccurate. In this case the inaccuracy seems to be due to the initial prior belief which are so different from the posterior ones.

4.4 Zeros Inferred by Generalized Belief Propagation

Belief propagation algorithms were extended yielding the class of *generalized belief propagation* (GBP) algorithms [Yedidia, Freeman, and Weiss 2000]. These algorithms fully process subparts of the networks, transforming it closer to a tree structure on which IBP can be more effective [Dechter, Mateescu, and Kask 2002; Mateescu, Kask, Gogate, and Dechter 2010]. The above results for IBP can now be extended to GBP and in particular to the variant of *iterative join-graph propagation*, IJGP [Dechter, Mateescu, and Kask 2002]. The algorithm applies message passing over a partition of the CPTs into clusters, called a join-graph, rather than over the dual graph. The set of clusters in such a partition defines a unique dual graph (i.e., each cluster is a node). This dual graph can be associated with various dual join-graphs, each defined by the labeling on the arcs between neighboring cluster nodes.

Algorithm IJGP has an accuracy parameter i, called i-bound, which restricts the maximum number of variables that can appear in a cluster and it is more accurate as i grows. The extension of all the previous observations regarding zeros to IJGP is straightforward and is summarized next, where the inferred approximation of the belief $P_{calB}(X_i|e)$ computed by IJGP is denoted by $P_{IJGP}(X_i|e)$.

THEOREM 21. *Given a belief network $\mathcal{B}$ to which IJGP is applied then:*

1. *IJGP generates all its $P_{IJGP}(x_i|e) = 0$ in finite time, that is, there exists a number k, such that no $P_{IJGP}(x_i) = 0$ will be generated after k iterations.*
2. *Whenever IJGP determines $P_{IJGP}(x_i|e) = 0$, it stays 0 during all subsequent iterations.*
3. *Whenever IJGP determines $P_{IJGP}(x_i|e) = 0$, then $Bel(x_i) = 0$.*

5 The Impact of IBP's Inferred Zeros

This section discusses the ramifications of having sound inferred zero beliefs.

5.1 The Inference Power of IBP

We now show that the inference power of IBP for zeros is sometimes very limited and other times strong, exactly wherever arc-consistency is weak or strong.

Cases of weak inference power. Consider the belief network described in Example 20. The flat constraint network of that belief network is defined over the scopes S_1=$\{H_1, X_1, X_2\}$, S_2=$\{H_2, X_2, X_3\}$, S_3=$\{H_3, X_1, X_3\}$. The constraints are defined by: R_{S_i} = $\{(1,1,2)$, $(1,2,1)$, $(1,3,3)$, $(0,1,1)$, $(0,1,3)$, $(0,2,2)$, $(0,2,3)$, $(0,3,1)$, $(0,3,2)\}$. The prior probabilities for X_i's imply unary constraints equal to the full domain {1,2,3}. An arc-minimal dual join-graph that is identical to the constraint network is given in Figure 4b. In this case, IBP-RDAC sends as messages the full domains of the variables and thus no tuple is removed from any constraint. Since IBP infers the same zeros as arc-consistency, IBP will also *not* infer any zeros. Since the true probability of most tuples is zero, we can conclude that the inference power of IBP on this example is weak or non-existent.

The weakness of arc-consistency in this example is not surprising. Arc-consistency is known to be far from complete. Since every constraint network can be expressed as a belief network (by adding a variable for each constraint as we did in the above example) and since arc-consistency can be arbitrarily weak on some constraint networks, so could be IBP.

Cases of strong inference power. The relationship between IBP and arc-consistency ensures that IBP is zero-complete, whenever arc-consistency is. In general, if for a flat constraint network of a belief network $\mathcal{B}$, arc-consistency removes all the inconsistent domain values, then IBP will also discover all the true zeros of $\mathcal{B}$. Examples of constraint networks that are complete for arc-consistency are max-closed constraints. These constraints have the property that if 2 tuples are in the relation so is their intersection. Linear constraints are often max-closed and so are Horn clauses (see [Dechter 2003]). Clearly, IBP is zero complete for acyclic networks which include binary trees, polytrees and networks whose dual graph is a hypertree [Dechter 2003]. This is not too illuminating though as we know that IBP is fully complete (not only for zeros) for such networks.

An interesting case is when the belief network has no evidence. In this case, the flat network always corresponds to the *causal constraint network* defined in [Dechter and Pearl 1991]. The inconsistent tuples or domain values are already explicitly described in each relation and no new zeros can be inferred. What is more interesting is that in the absence of evidence IBP is also complete for non-zero beliefs for many variables as we show later.

5.2 IBP and Loop-Cutset

It is well-known that if evidence nodes form a loop-cutset, then we can transform any multiply-connected belief network into an equivalent singly-connected network

which can be solved by belief propagation, leading to the loop-cutset conditioning method [Pearl 1988]. Now that we established that inferred zeros, and in particular inferred evidence (i.e., when only a single value in the domain of a variable has a non-zero probability) are sound, we show that evidence play the cutset role automatically during IBP's performance. Indeed, we can show that during IBP's operation, an observed node X_i in a Bayesian network blocks the path between its parents and its children as defined in the d-separation criteria. All the proofs of claims appearing in Section 5.2 and Section 5.3 can be found in [Bidyuk and Dechter 2001].

PROPOSITION 22. *Let X_i be an observed node in a belief network $\mathcal{B}$. Then for any child Y_j of node X_i, the belief of Y_j computed by IBP is not dependent on the messages that X_i receives from its parents $pa(X_i)$ or the messages that node X_i receives from its other children $Y_k, k \neq j$.*

From this we can conclude that:

THEOREM 23. *If evidence nodes, original or inferred, constitute a loop-cutset, then IBP converges to the correct beliefs in linear time.*

5.3 IBP on Irrelevant Nodes

An orthogonal property is that unobserved nodes that have only unobserved descendents are irrelevant to the beliefs of the remaining nodes and therefore, processing can be restricted to the relevant subgraphs. In IBP, this property is expressed by the fact that irrelevant nodes send messages to their parents that equally support each value in the domain of a parent and thus do not affect the computation of marginal posteriors of its parents.

PROPOSITION 24. *Let X_i be an unobserved node without observed descendents in $\mathcal{B}$ and let $\mathcal{B}'$ be a subnetwork obtained by removing X_i and its descendents from $\mathcal{B}$. Then, $\forall Y \in \mathcal{B}'$ the belief of Y computed by IBP over $\mathcal{B}$ equals the belief of Y computed by IBP over $\mathcal{B}'$.*

Thus, in a loopy network without evidence, IBP always converges after 1 iteration since only propagation of top-down messages affects the computation of beliefs and those messages do not change. Also in that case, IBP converges to the correct marginals for any node X_i such that there exists only one directed path from any ancestor of X_i to X_i. This is because the relevant subnetwork that contains only the node and its ancestors is singly-connected and by Proposition 24 they are the same as the beliefs computed by applying IBP to the complete network. In summary,

THEOREM 25. *Let $\mathcal{B}'$ be a subnetwork obtained from $\mathcal{B}$ by recursively eliminating all its unobserved leaf nodes. If observed nodes constitute a loop-cutset of $\mathcal{B}'$, then IBP applied to $\mathcal{B}$ converges to the correct beliefs for all nodes in $\mathcal{B}'$.*

THEOREM 26. *If a belief network does not contain any observed nodes or only has observed root nodes, then IBP always converges.*

In summary, in Sections 5.2 and 5.3 we observed that IBP exploits the two prop-

erties of observed and unobserved nodes, *automatically*, without any outside intervention for network transformation. As a result, the correctness and convergence of IBP on a node X_i in a multiply-connected belief network will be determined by the structure restricted to X_i's relevant subgraph. If the relevant subnetwork of X_i is singly-connected relative to the evidence (observed or inferred), IBP will converge to the correct beliefs for node X_i.

6 Experimental Evaluation

The goal of the experiments is two-fold. First, since zero values inferred by IBP/IJGP are proved correct, we want to explore the behavior of IBP/IJGP for near zero inferred beliefs. Second, we want to explore the hypothesis that the loop-cutset impact on IBP's performance, as discussed in Section 5.2, also extends to variables with extreme support. The next two subsections are devoted to these two issues, respectively.

6.1 On the Accuracy of IBP in Near Zero Marginals

We test the performance of IBP and IJGP both on cases of strong and weak inference power. In particular, we look at networks where probabilities are extreme and investigate empirically the accuracy of IBP/IJGP across the range of belief values from 0 to 1. Since zero values inferred by IBP/IJGP are proved correct, we focus especially on the behavior of IBP/IJGP for near zero inferred beliefs.

Using names inspired by the well known measures in information retrieval, we report *Recall Absolute Error* and *Precision Absolute Error* over small intervals spanning $[0, 1]$. *Recall* is the absolute error averaged over all the exact beliefs that fall into the interval, and can therefore be viewed as capturing the level of completeness. For *precision*, the average is taken over all the belief values computed by IBP/IJGP that fall into the interval, and can be viewed as capturing soundness.

The X coordinate in Figure 5 and Figure 10 denotes the interval $[X, X + 0.05)$. For the rest of the figures, the X coordinate denotes the interval $(X-0.05, X]$, where the 0 interval is $[0, 0]$. The left Y axis corresponds to the histograms (the bars), while the right Y axis corresponds to the absolute error (the lines). For problems with binary variables, we only show the interval [0, 0.5] because the graphs are symmetric around 0.5. The number of variables, number of evidence variables and induced width w* are reported in each graph.

Since the behavior within each benchmark is similar, we report a subset of the results (for an extended report see [Rollon and Dechter 2009].

Coding networks. Coding networks are the famous case where IBP has impressive performance. The instances are from the class of linear block codes, with 50 nodes per layer and 3 parent nodes for each variable. We experiment with instances having three different values of channel noise: 0.2, 0.4 and 0.6. For each channel value, we generate 1000 samples.

Figure 5 shows the results. When the noise level is 0.2, all the beliefs computed

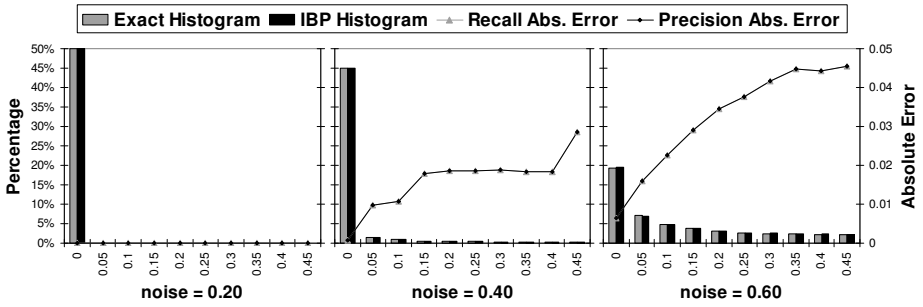


Figure 5. Coding, N=200, evidence=100, w*=15, 1000 instances.

by IBP are extreme. The Recall and Precision are very small, of the order of 10^{-11}. So, in this case, all the beliefs are very small (i.e., ϵ small) and IBP is able to infer them correctly, resulting in almost perfect accuracy (IBP is indeed perfect in this case for the bit error rate). As noise increases, the Recall and Precision get closer to a bell shape, indicating higher error for values close to 0.5 and smaller error for extreme values. The histograms show that fewer belief values are extreme as noise increases.

Linkage Analysis networks. Genetic linkage analysis is a statistical method for mapping genes onto a chromosome. The problem can be modeled as a belief network. We experimented with four *pedigree* instances from the UAI08 competition. The domain size ranges between 1 to 4. For these instances exact results are available. Figure 6 shows the results. We observe that the number of exact 0 beliefs is small and IJGP correctly infers all of them. The behavior of IJGP for ϵ small beliefs varies accross instances. For *pedigree1*, the Exact and IJGP histograms are about the same (for all intervals). Moreover, Recall and Precision errors are relatively small. For the rest of the instances, the accuracy of IJGP for extreme inferred marginals decreases. Notice that IJGP infers more ϵ small beliefs than the number of exact extremes in the corresponding intervals, leading to relatively high Precision error while small Recall error. The behaviour for beliefs in the 0.5 interval is reversed, leading to high Recall error while small Precision error. As expected, the accuracy of IJGP improves as the value of the control parameter i-bound increases.

Grid networks. Grid networks are characterized by two parameters (N, D), where $N \times N$ is the size of the network and D is the percentage of determinism (i.e., the percentage of values in all CPTs assigned to either 0 or 1). We experiment with *grids2* instances from the UAI08 competition. They are characterized by parameters $(\{16, \ldots, 42\}, \{50, 75, 90\})$. For each parameter configuration, there are samples of size 10 generated by randomly assigning value 1 to one leaf node.

Figure 7 and Figure 8 report the results. IJGP correctly infers all 0 beliefs.

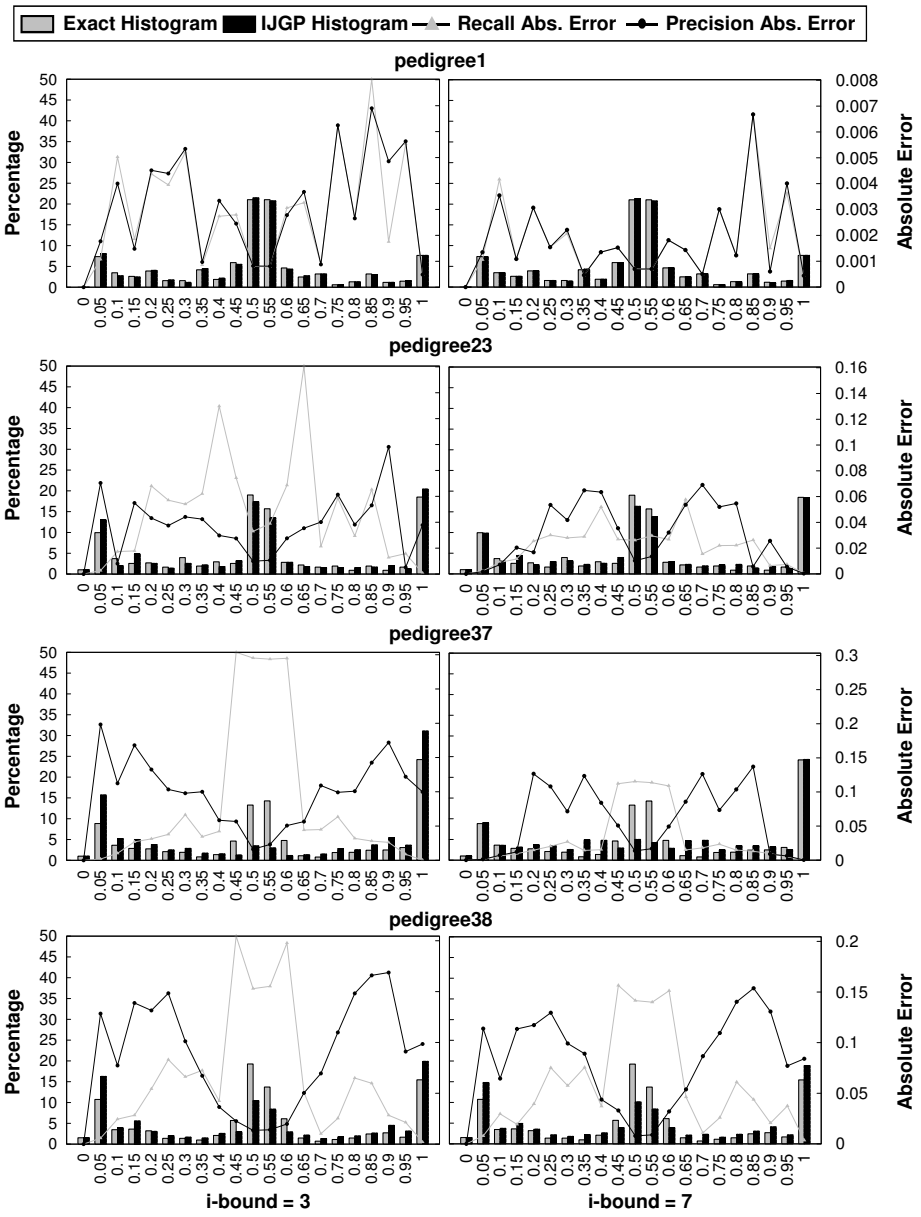


Figure 6. Results on pedigree instances. Each row is the result for one instance. Each column is the result of running IJGP with i-bound equal to 3 and 7, respectively. The number of variables N, number of evidence variables NE, and induced width w* of each instance is as follows. Pedigree1: $N = 334$, $NE = 36$ and w*=21; pedigree23: $N = 402$, $NE = 93$ and w*=30; pedigree37: $N = 1032$, $NE = 306$ and w*=30; pedigree38: $N = 724$, $NE = 143$ and w*=18.

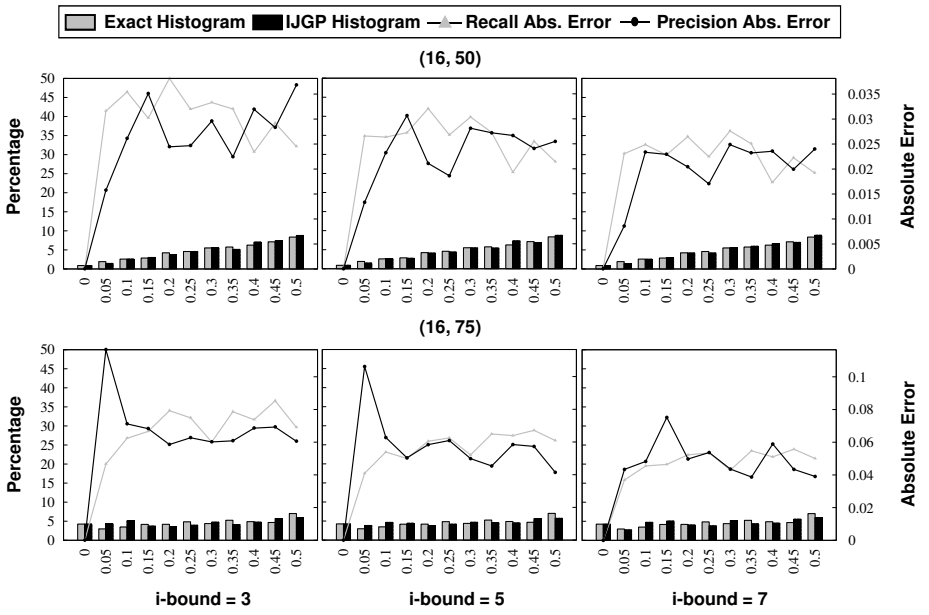


Figure 7. Results on grids2 instances. First row shows the results for parameter configuration $(16, 50)$. Second row shows the results for $(16, 75)$. Each column is the result of running IJGP with i-bound equal to 3, 5, and 7, respectively. Each plot indicates the mean value for up to 10 instances. Both parameter configurations have 256 variables, one evidence variable, and induced width w*=22.

However, its performance for ϵ small beliefs is quite poor. Only for networks with parameters $(16, 50)$ the Precision error is relatively small (less than 0.05). If we fix the size of the network and the i-bound, both Precision and Recall errors increase as the determinism level D increases. The histograms clearly show the gap between the number of true ϵ small beliefs and the ones inferred by IJGP. As before, the accuracy of IJGP improves as the value of the control parameter i-bound increases.

Two-layer noisy-OR networks. Variables are organized in two layers where the ones in the second layer have 10 parents. Each probability table represents a noisy OR-function. Each parent variable y_j has a value $P_j \in [0..P_{noise}]$. The CPT for each variable in the second layer is then defined as, $P(x = 0|y_1, \ldots, y_P) = \prod_{y_j=1} P_j$ and $P(x = 1|y_1, \ldots, y_P) = 1 - P(x = 0|y_1, \ldots, y_P)$. We experiment on *bn2o* instances from the UAI08 competition.

Figure 9 reports the results for 3 instances. In this case, IJGP is very accurate for all instances. In particular, the accuracy in ϵ small beliefs is very high.

CPCS networks. These are medical diagnosis networks derived from the Computer-Based Patient Care Simulation system (CPCS) expert system. We tested on two

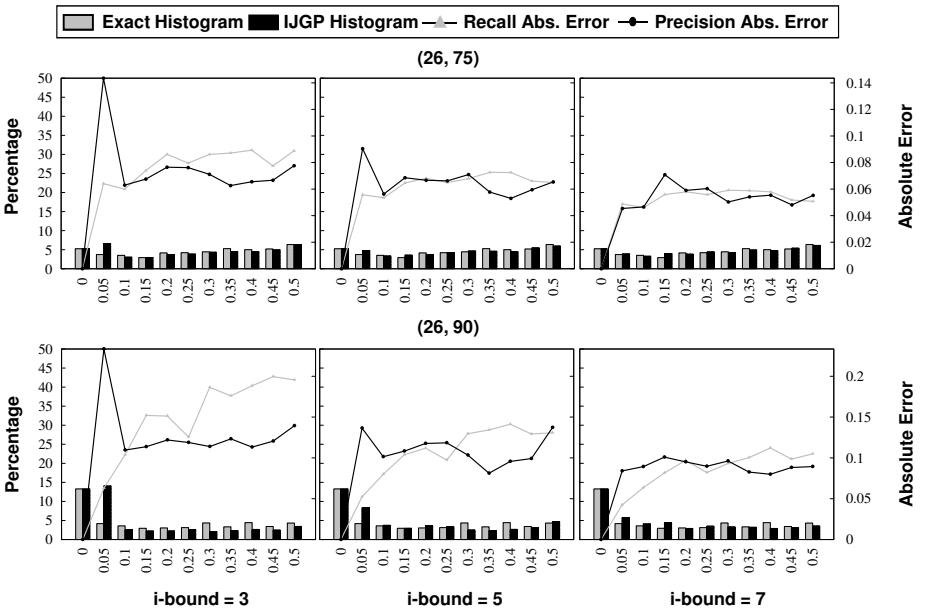


Figure 8. Results on grids2 instances. First row shows the results for parameter configuration $(26, 75)$. Second row shows the results for $(26, 90)$. Each column is the result of running IJGP with i-bound equal to 3, 5 and 7, respectively. Each plot indicates the mean value for up to 10 instances. Both parameter configurations have 676 variables, one evidence variable, and induced width w*=40.

networks, *cpcs54* and *cpcs360*, with 54 and 360 variables, respectively. For the first network, we generate samples of size 100 by randomly assigning 10 variables as evidence. For the second network, we also generate samples of the same size by randomly assigning 20 and 30 variables as evidence.

Figure 10 shows the results. The histograms show opposing trends in the distribution of beliefs. Although irregular, the absolute error tends to increase towards 0.5 for cpcs54. In general, the error is quite small throughout all intervals and, in particular, for inferred extreme marginals.

6.2 On the Impact of Epsilon Loop-Cutset

In [Bidyuk and Dechter 2001] we explored also the hypothesis that the loop-cutset impact on IBP's performance, as discussed in Section 5.2, extends to variables with extreme support. Extreme support is expressed in the form of either extreme prior value $P(x_i) < \epsilon$ or strong correlation with an observed variable. We hypothesize that a variable X_i with extreme support nearly-cuts the information flow from its parents to its children similar to an observed variable. Subsequently, we conjecture that when a subset of variables with extreme support, called ϵ-*cutset*, form a loop-

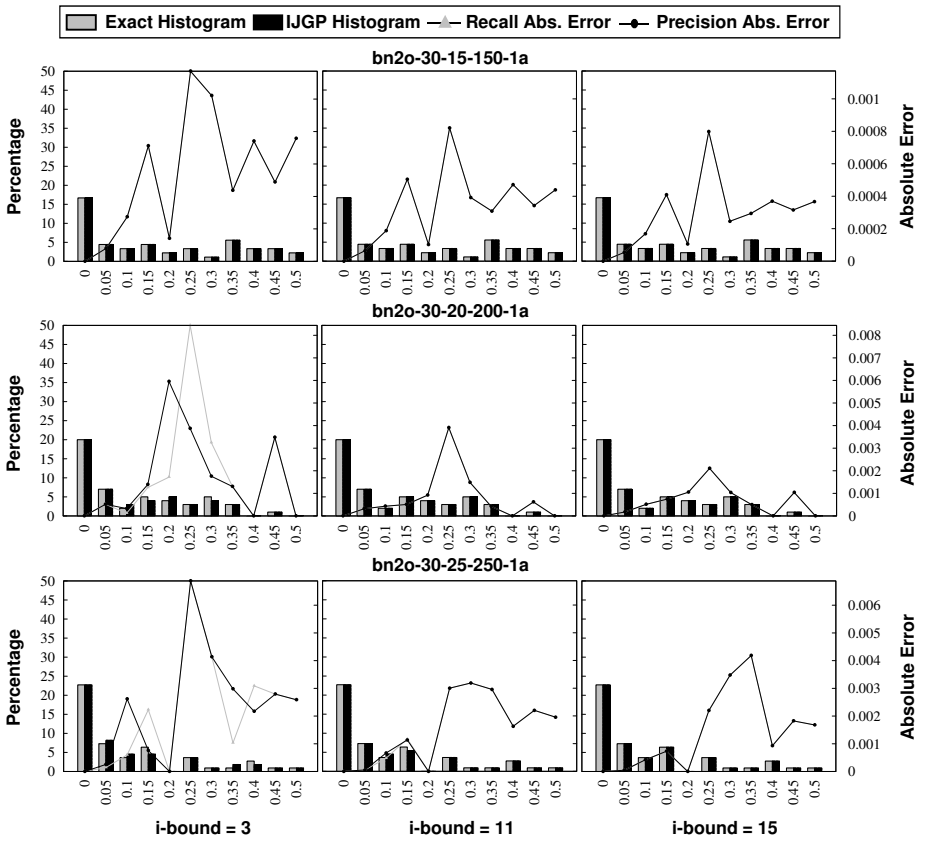


Figure 9. Results on bn2o instances. Each row is the result for one instance. Each column in each row is the result of running IJGP with i-bound equal to 3, 5 and 7, respectively. The number of variables N, number of evidence variables NE, and induced width w* of each instance is as follows. bn2o-30-15-150-1a: $N = 45$, $NE = 15$, and w*=24; bn2o-30-20-200-1a: $N = 50$, $NE = 20$, and w*=27; bn2o-30-25-250-1a: $N = 55$, $NE = 25$, and w*=26.

cutset of the graph, IBP converges and computes beliefs that approach exact ones.

We will briefly recap the empirical evidence supporting the hypothesis in 2-layer noisy-OR networks. The number of root nodes m and total number of nodes n was fixed in each test set (indexed $m - n$). Generating the networks, each leaf node Y_j was added to the list of children of a root node U_i with probability 0.5. All nodes were bi-valued. All leaf nodes were observed. We used average absolute error in the posterior marginals (averaged over all unobserved variables) to measure IBP's accuracy and the percent of variables for which IBP converged as a measure of convergence. In each group of experiments, the results were averaged over 100 instances.

In one set of experiments, we measured the performance of IBP while changing

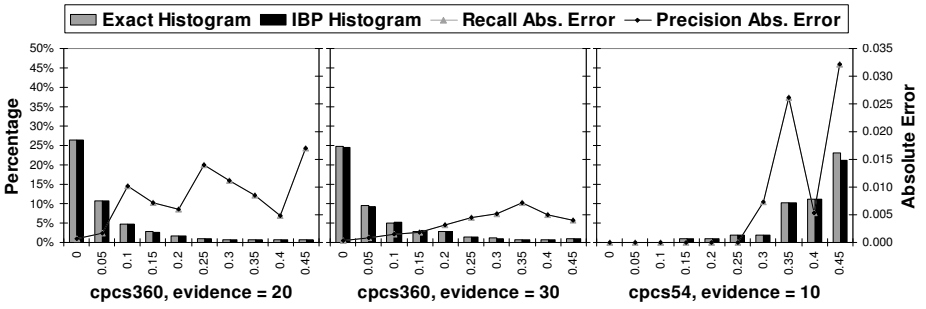


Figure 10. CPCS54, 100 instances, w*=15; CPCS360, 5 instances, w*=20

the number of observed loop-cutset variables (we fixed all priors to $(.5, .5)$ and picked observed value for loop-cutset variables at random). The results are shown in Figure 11, top. As expected, the number of converged nodes increased and the absolute average error decreased monotonically as number of observed loop-cutset nodes increased.

Then, we repeated the experiment except now, instead of instantiating a loop-cutset variable, we set its priors to extreme (ϵ, 1-ϵ) with ϵ=$1E-10$, i.e., instead of increasing the number of observed loop-cuset variables, we increased the number of ϵ-cutset variables. If our hypothesis is correct, increasing the size of ϵ-cutset should produce an effect similar to increasing the number of observed loop-cutset variables, namely, improved convergence and better accuracy in IBP computed beliefs. The results, in Figure 11, bottom, demonstrate that initially, as the number of ϵ-cutset variables grows, the performance of IBP improves just as we conjectured. However, the percentage of nodes with converged beliefs never reaches 100% just like the average absolute error converges to some $\delta > 0$. In the case of 10-40 network, the number of converged beliefs (average absolute error) reaches maximum of $\approx$ 95% (minimum of $\approx$ .001) at 3 ϵ-cutset nodes and then drops to $\approx$ 80% (increases to $\approx$ .003) as the size of ϵ-cutset increases.

To further investigate the effect of the *strength* of ϵ-support on the performance of IBP, we experimented on the same 2-layer networks varying the prior values of the loop-cutset nodes from (ϵ, 1-ϵ) to (1-ϵ, ϵ) for $\epsilon \in [1E-10, .5]$. As shown in Figure 12, initially, as ϵ decreased, the convergence and accuracy of IBP worsened. This effect was previously reported by Murphy, Weiss, and Jordan [Murphy, Weiss, and Jordan 2000]. However, as the priors of loop-cutset nodes continue to approach 0 and 1, the average error value approaches 0 and the number of converged nodes reaches 100%. Note that convergence is not symmetric with respect to ϵ. The average absolute error and percentage of converged nodes approach 0 and 1 respectively for ϵ=1-(1E-10) but not for ϵ=1E-10 (which we also observed in Figure 11, bottom).

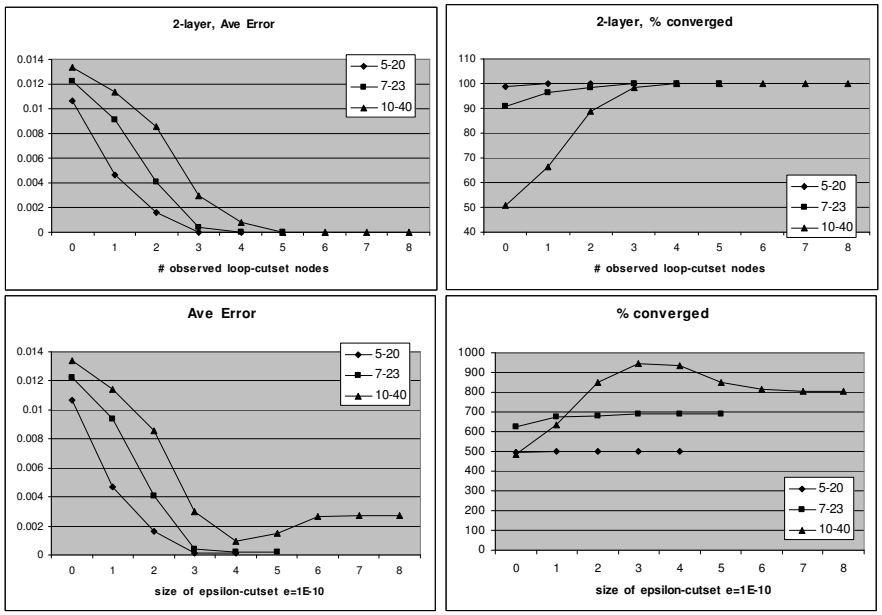


Figure 11. Results for 2-layer Noisy-OR networks. The average error and the number of converged nodes vs the number of truly observed loop-cutset nodes (top) and the size of of ϵ-cutset (bottom).

7 Conclusion

The paper provides insight into the power of the Iterative Belief Propagation (IBP) algorithm by making its relationship with constraint propagation explicit. We show that the power of belief propagation for zero beliefs is identical to the power of arc-consistency in removing inconsistent domain values. Therefore, the strength and weakness of this scheme can be gleaned from understanding the inference power of arc-consistency. In particular we show that the inference of zero beliefs (marginals) by IBP and IJGP is always sound. These algorithms are guaranteed to converge for inferred zeros and are as efficient as the corresponding constraint propagation algorithms.

Then the paper empirically investigates whether the sound inference of zeros by IBP is extended to near zeros. We show that while the inference of near zeros is often quite accurate, it can sometimes be extremely inaccurate for networks having significant determinism. Specifically, for networks without determinism IBP's near zero inference was sound in the sense that the average absolute error was contained within the length of the 0.05 interval (see *two layer noisy-OR* and *CPCS* benchmarks). However, the behavior was different on benchmark networks having determinism. For example, experiments on *coding* networks show that IBP is almost perfect, while for *pedigree* and *grid* networks the results are quite inaccurate

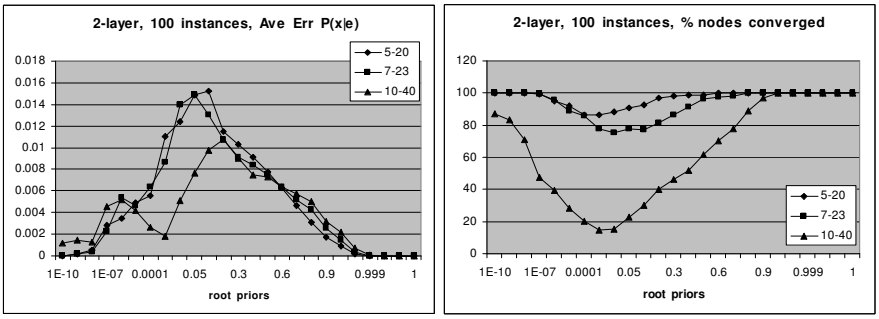


Figure 12. Results for 2-layer Noisy-OR networks. The average error and the percent of converged nodes vs ϵ-support.

near zeros.

Finally, we show that evidence, observed or inferred, automatically acts as a cycle-cutting mechanism and improves the performance of IBP. We also provide preliminary empirical evaluation showing that the effect of loop-cutset on the accuracy of IBP extends to variables that have extreme probabilities.

Acknowledgement: This work was supported in part by NSF grant IIS-0713118 and by NIH grant 5R01HG004175-03.

References

Bidyuk, B. and R. Dechter (2001). The epsilon-cutset effect in Bayesian networks, r97, r97a in http://www.ics.uci.edu/ dechter/publications. Technical report, University of California, Irvine.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann Publishers.

Dechter, R. and R. Mateescu (2003). A simple insight into iterative belief propagation's success. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pp. 175–183.

Dechter, R., R. Mateescu, and K. Kask (2002). Iterative join-graph propagation. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pp. 128–136.

Dechter, R. and J. Pearl (1991). Directed constraint networks: A relational framework for causal reasoning. In *Proceedings of the Twelfth International Joint Conferences on Artificial Intelligence (IJCAI'91)*, pp. 1164–1170.

Ihler, A. T. (2007). Accuracy bounds for belief propagation. In *Proceedings of the Twenty Third Conference on Uncertainty in Artificial Intelligence (UAI'07)*.

Ihler, A. T., J. W. Fisher, III, and A. S. Willsky (2005). Loopy belief propagation: Convergence and effects of message errors. *J. Machine Learning Research 6*, 905–936.

Koller, D. (2010). Belief propagation in loopy graphs. In *Heuristics, Probabilities and Causality: A tribute to Judea Pearl, Editors, R. Dechter, H. Gefner and J. Halpern*.

Mackworth, A. K. (1977). Consistency in networks of relations. *Artificial Intelligence 8*(1), 99–118.

Mateescu, R., K. Kask, V. Gogate, and R. Dechter (2010). Iterative join-graph propagation. *Journal of Artificial Intelligence Research (JAIR) (accepted, 2009)*.

McEliece, R. J., D. J. C. MacKay, and J. F. Cheng (1998). Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication 16*(2), 140–152.

Mooij, J. M. and H. J. Kappen (2007). Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. Information Theory 53*(12), 4422–4437.

Mooij, J. M. and H. J. Kappen (2009). Bounds on marginal probability distributions. In *Advances in Neural Information Processing Systems 21 (NIPS'08)*, pp. 1105–1112.

Murphy, K., Y. Weiss, and M. Jordan (2000). Loopy-belief propagation for approximate inference: An empirical study. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pp. 467–475.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial. Intelligence 29*(3), 241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers.

Rish, I., K. Kask, and R. Dechter (1998). Empirical evaluation of approximation algorithms for probabilistic decoding. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*, pp. 455–463.

Rollon, E. and R. Dechter (December, 2009). Some new empirical analysis in iterative join-graph propagation, r170 in http://www.ics.uci.edu/ dechter/publications. Technical report, University of California, Irvine.

Roosta, T. G., M. J. Wainwright, and S. S. Sastry (2008). Convergence analysis of reweighted sum-product algorithms. *IEEE Trans. Signal Processing 56*(9), 4293–4305.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2000). Generalized belief propagation. In *Advances in Neural Information Processing Systems 13 (NIPS'00)*, pp. 689–695.

10

Bayesian Nonparametric Learning: Expressive Priors for Intelligent Systems

Michael I. Jordan

1 Introduction

One of the milestones in the development of artificial intelligence (AI) is the embrace of uncertainty and inductive reasoning as primary concerns of the field. This embrace has been a surprisingly slow process, perhaps because the naive interpretation of "uncertain" seems to convey an image that is the opposite of "intelligent." That the field has matured beyond this naive opposition is one of the singular achievements of Judea Pearl. While the pre-Pearl AI researcher tended to focus on mimicking the deductive capabilities of human intelligence, a post-Pearl researcher has been sensitized to the inevitable uncertainty that intelligent systems face in any realistic environment, and the need to explicitly represent that uncertainty so as to be able to mitigate its effects. Not only does this embrace of uncertainty accord more fully with the human condition, but it also recognizes that the first artificially intelligent systems—necessarily limited in their cognitive capabilities—will be if anything *more* uncertain regarding their environments than us humans. It is only by embracing uncertainty that a bridge can be built from systems of limited intelligence to those having robust human-level intelligence.

A computational perspective on uncertainty has two aspects: the explicit representation of uncertainty and the algorithmic manipulation of this representation so as to transform and (often) to reduce uncertainty. In his seminal 1988 book, *Probabilistic Reasoning in Intelligent Systems*, Pearl showed that these aspects are intimately related. In particular, obtaining a compact representation of uncertainty has important computational consequences, leading to efficient algorithms for marginalization and conditioning. Moreover, marginalization and conditioning are the core inductive operations that tend to reduce uncertainty. Thus, by developing an effective theory of the representation of uncertainty, Pearl was able to also develop an effective computational approach to probabilistic reasoning.

Uncertainty about an environment can also be reduced by simply observing that environment; i.e., by learning from data. Indeed, another response to the early focus on deduction in AI has been to emphasize learning as a pathway to the development of intelligent systems. In the 1980's, concurrently with Pearl's work on probabilistic expert systems, this perspective was taken up in earnest, building on an earlier tradition in pattern recognition (which itself built on even earlier traditions in statis-

tics). The underlying inductive principle was essentially the law of large numbers, a principle of probability theory which states that the statistical aggregation of independent, identically distributed samples yields a decrease of uncertainty that goes (roughly speaking) at a rate inversely proportional to the square root of the number of samples. The question has been how to perform this "aggregation," and the learning field has been avidly empirical, exploring a variety of computational architectures, including extremely simple representations (e.g., nearest neighbor), ideas borrowed from deductive traditions (e.g., decision trees), ideas closely related to classical statistical models (e.g., boosting and the support vector machine), and architectures motivated at least in part by complex biological and physical systems (e.g., neural networks). Several of these architectures have factorized or graphical representations, and numerous connections to graphical models have been made.

A narrow reader of Pearl's book might wish to argue that learning is not distinct from the perspective on reasoning presented in that book; in particular, observing the environment is simply a form of conditioning. This perspective on learning is indeed reasonable if we assume that a learner maintains an explicit probabilistic model of the environment; in that case, making an observation merely involves instantiating some variable in the model. However, many learning researchers do not wish to make the assumption that the learner maintains an explicit probabilistic model of the environment, and many algorithms developed in the learning field involve some sort of algorithmic procedure that is not necessarily interpretable as computing a conditional probability. These procedures are instead justified in terms of their unconditional performance when used again and again on various data sets.

Here we are of course touching on the distinction between the Bayesian and the frequentist approaches to statistical inference. While this is not the place to develop that distinction in detail, it is worth noting that statistics—the field concerned with the theory and practice of inference—involves the interplay of the conditional (Bayesian) and the unconditional (frequentist) perspectives and this interplay also underlies many developments in AI research. Indeed, the trend since Pearl's work in the 1980's has been to blend reasoning and learning: put simply, one does not need to learn (from data) what one can infer (from the current model). Moreover, one does not need to infer what one can learn (intractable inferential procedures can be circumvented by collecting data). Thus learning (whether conditional or not) and reasoning interact. The most difficult problems in AI are currently being approached with methods that blend reasoning with learning. While the extremes of classical expert systems and classical tabula rasa learning are still present and still have their value in specialized situations, they are not the centerpieces of the field. Moreover, the caricatures of probabilistic reasoning and statistical inference that fed earlier ill-informed debates in AI have largely vanished. For this we owe much to Judea Pearl.

There remain, however, a number of limitations—both perceived and real—of probabilistic and statistical approaches to AI. In this essay, I wish to focus on some

of these limitations and provide some suggestions as to the way forward.

It is both a perception and reality that to use probabilistic methods in AI one is generally forced to write down long lists of assumptions. This is often a helpful exercise, in that it focuses a designer to bring hidden assumptions to the foreground. Moreover, these assumptions are often qualitative in nature, with the quantitative details coming from elicitation methods (i.e., from domain experts) and learning methods. Nonetheless, the assumptions are not always well motivated. In particular, independence assumptions are often imposed for reasons of computational convenience, not because they are viewed as being true, and the effect on inference is not necessarily clear. More subtly, and thus of particular concern, is the fact that the tail behavior of probability distributions is often not easy to obtain (from elicitation or from data), and choices of convenience are often made.

A related issue is that probabilistic methods are often not viewed as sufficiently expressive. One common response to this issue has involved trying to bring ideas from first-order logic to bear on probabilistic modeling. This line of work has, however, mainly involved using logical representations as a high-level interface for model specification and then compiling these representations down to flat probabilistic representations for inference. It is not yet clear how to bring together the powerful inferential methods of logic and probability into an effective computational architecture.

In the current paper, we will pursue a different approach to expressive probabilistic representation and to a less assumption-laden approach to inference. The idea is to move beyond the simple fixed-dimensional random variables that have been generally used in graphical models (multinomials, Gaussians and other exponential family distributions) and to consider a wider range of probabilistic representations. We are motivated by the ubiquity of flexible data structures in computer science—the field is based heavily on objects such as trees, lists and collections of sets that are able to expand and contract as needed. Moreover, these data structures are often associated with combinatorial and algebraic identities that lead to efficient algorithms. We would like to mimic this flexibility within the world of probabilistic representations.

In fact, the existing field of *stochastic processes* provides essentially this kind of flexibility. Recall that a stochastic process is an indexed collection of random variables, where the index set can be infinite (countably infinite or uncountably infinite) [Karlin and Taylor 1975]. Within the general theory of stochastic processes it is quite natural to define probability distributions on objects such trees, lists and collections of sets. It is also possible to define probability distributions on spaces of probability distributions, yielding an appealing recursivity. Moreover, many stochastic processes have interesting ties to combinatorics (and to other areas of mathematics concerned with compact structure, such as algebra). Probability theorists have spent many decades developing these ties and a rich literature on "combinatorial stochastic processes" has emerged [Pitman 2002]. It is natural to

take this literature as a point of departure for the development of expressive data structures for computationally efficient reasoning and learning.

One general way to use stochastic processes in inference is to take a Bayesian perspective and replace the parametric distributions used as priors in classical Bayesian analysis with stochastic processes. Thus, for example, we could consider a model in which the prior distribution is a stochastic process that ranges over trees of arbitrary depth and branching factor. Combining this prior with a likelihood, we obtain a posterior distribution that is also a stochastic process that ranges over trees of arbitrary depth and branching factor. Bayesian learning amounts to updating one flexible representation (the *prior stochastic process*) into another flexible representation (the *posterior stochastic process*).

This idea is not new, indeed it is the core idea in an area of research known as *Bayesian nonparametrics*, and there is a small but growing community of researchers who work in the area. The word "nonparametrics" needs a bit of explanation. The word does not mean "no parameters"; indeed, many stochastic processes can be usefully viewed in terms of parameters (often, infinite collections of parameters). Rather, it means "not parametric," in the sense that Bayesian nonparametric inference is not restricted to objects whose dimensionality stays fixed as more data is observed. The spirit of Bayesian nonparametrics is that of flexible data structures—representations can grow as needed. Moreover, stochastic processes yield a much broader class of probability distributions than the class of exponential family distributions that is the focus of the graphical model literature. In this sense, Bayesian nonparametric learning is less assumption-laden than classical Bayesian parametric learning.

In this paper we offer an invitation to Bayesian nonparametrics. Our presentation is meant to evoke Pearl's presentation of Bayesian networks in that our focus is on foundational representational issues. As in the case of graphical models, if the representational issues are handled well, then there are favorable algorithmic consequences. Indeed, the parallel is quite strong—in the case of graphical models, these algorithmic consequences are combinatorial in nature (they involve the combinatorics of sums and products), and in the case of Bayesian nonparametrics favorable algorithmic consequences also arise from the combinatorial properties of certain stochastic process priors.

2 De Finetti's theorem and the foundations of Bayesian inference

A natural point of departure for our discussion is a classical theorem due to Bruno De Finetti that is one of the pillars of Bayesian inference. This core result not only suggests the need for prior distributions in statistical models but it also leads directly to the consideration of stochastic processes as Bayesian priors.

Consider an infinite sequence of random variables, $(X_1, X_2, \ldots)$. To simplify our discussion somewhat, let us assume that these random variables are discrete. We say

that such a sequence is *infinitely exchangeable* if the joint probability distribution of any finite subset of those random variables is invariant to permutation. That is, for any N, we have $p(x_1, x_2, \ldots, x_N) = p(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(N)})$, where π is a permutation and p is a probability mass function. De Finetti's theorem states that $(X_1, X_2, \ldots)$ are infinitely exchangeable if and only the joint probability distribution of any finite subset can be written as a marginal probability in the following way:

$$p(x_1, x_2, \ldots, x_N) = \int \prod_{i=1}^{N} p(x_i \,|\, G) P(dG). \tag{1}$$

In one direction this theorem is straightforward: If the joint distribution can be written as in integral in this way, then we clearly have invariance to permutation (because the product is invariant to permutation). It is the other direction that is non-trivial. It states that for exchangeable random variables, there necessarily exists an underlying random element G, and a probability distribution P, such that the random variables X_i are conditionally independent given G, and such that their joint distribution is obtained by integrating over the distribution P. If we view G as a "parameter," then this theorem can be interpreted as stating that exchangeability implies the existence of an underlying parameter and a prior distribution on that parameter. As such, De Finetti's theorem is often viewed as providing foundational support for the Bayesian paradigm.

We placed "parameter" in quotes in the preceding paragraph because there is no restriction that G should be a finite-dimensional object. Indeed, the full import of De Finetti's theorem is clear when we realize that in many instances G is in fact an infinite-dimensional object, and P defines a stochastic process.

Let us give a simple example. The *Pólya urn model* is a simple probability model for sequentially labeling the balls in an urn. Consider an empty urn and a countably infinite collection of colors. Pick a color at random according to some fixed distribution G_0 and place a ball having that color in the urn. For all subsequent balls, either choose a ball from the urn (uniformly at random) and return that ball to the urn with another ball of the same color, or choose a new color from G_0 and place a ball of that color in the urn. Mathematically, we have:

$$p(X_i = k \,|\, x_1, \ldots x_{i-1}) \propto \begin{cases} n_k & \text{if } x_j = k \text{ for some } j \in \{1, \ldots, i-1\} \\ \alpha_0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\alpha_0 > 0$ is a parameter of the process.

It turns out that the Pólya urn model is exchangeable. That is, even though we defined the model by picking a particular ordering of the balls, the resulting distribution is independent of the order. This is proved by writing the joint distribution $p(x_1, x_2, \ldots, x_N)$ as a product of conditionals of the form in Eq. (2) and noting (after some manipulation) that the resulting expression is independent of order.

While the Pólya urn model defines a distribution on labels, it can also be used to induce a distribution on partitions. This is achieved by simply partitioning the balls

into groups that have the same color. This distribution on partitions is known as the *Chinese restaurant process* [Aldous 1985]. As we discuss in more detail in Section 4, the Chinese restaurant process and the Pólya urn model can be used as the basis of a Bayesian nonparametric model of clustering where the random partition provides a prior on clusterings and the color associated with a given cell can be viewed as a parameter vector for a distribution associated with a given cluster.

The exchangeability of the Pólya urn model implies—by De Finetti's theorem—the existence of an underlying random element G that renders the ball colors conditionally independent. This random element is not a classical fixed-dimension random variable; rather, it is a stochastic process known as the *Dirichlet process*. In the following section we provide a brief introduction to the Dirichlet process.

3 The Dirichlet process

In thinking about how to place random distributions on infinite objects, it is natural to begin with the special case of the positive integers. A distribution $\pi = (\pi_1, \pi_2, \ldots)$ on the integers can be viewed as a sequence of nonnegative numbers that sum to one. How can we obtain *random* sequences that sum to one?

One solution to this problem is provided by a procedure known as "stick-breaking." Define an infinite sequence of independent random variables as follows:

$$\beta_k \sim \text{Beta}(1, \alpha_0) \qquad\qquad k = 1, 2, \ldots, \tag{3}$$

where $\alpha_0 > 0$ is a parameter. Now define an infinite random sequence as follows:

$$\pi_1 = \beta_1, \qquad\qquad \pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) \qquad\qquad k = 2, 3, \ldots. \tag{4}$$

It is not difficult to show that $\sum_{k=1}^{\infty} \pi_k = 1$ (with probability one).

We can exploit this construction to generate a large class of random distributions on sets other than the integers. Consider an arbitrary measurable space Ω and let G_0 be a probability distribution on Ω. Draw an infinite sequence of points $\{\phi_k\}$ independently from G_0. Now define:

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}, \tag{5}$$

where δ_{ϕ_k} is a unit mass at the point ϕ_k. Clearly G is a measure. Indeed, for any measurable subset B of Ω, $G(A)$ just adds up the values π_k for those k such that $\phi_k \in B$, and this process satisfies the countable additivity needed in the definition of a measure. Moreover, G is a probability measure, because $G(\Omega) = 1$.

Note that G is random in two ways—the weights π_k are obtained by a random process, and the locations ϕ_k are also obtained by a random process. While it seems clear that such an object is not a classical finite-dimensional random variable, in

what sense is G is a stochastic process; i.e., an indexed collection of random variables? The answer is that G is a stochastic process where the indexing variables are the measurable subsets of Ω. Indeed, for any fixed $A \subseteq \Omega$, $G(A)$ is a random variable. Moreover (and this is not an obvious fact), ranging over sets of subsets, $\{A_1, A_2, \ldots, A_K\}$, the joint distributions on the collections of random variables $\{G(A_i)\}$ are consistent with each other. This shows, via an argument in the spirit of the Kolmogorov theorem that G is a stochastic process. A more concrete understanding of this fact can be obtained by specializing to sets $\{A_1, A_2, \ldots, A_K\}$ that form a partition of Ω. In this case, the random vector $(G(A_1), G(A_2), \ldots, G(A_K))$ can be shown to have a classical finite-dimensional Dirichlet distribution:

$$(G(A_1), \ldots, G(A_K)) \sim \mathrm{Dir}(\alpha_0 G_0(A_1), \ldots, \alpha_0 G_0(A_K)), \tag{6}$$

from which the needed consistency properties follow immediately from classical properties of the Dirichlet distribution. For this reason, the stochastic process defined by Eq. (5) is known as a *Dirichlet process*. Eq. (6) can be summarized as saying that a Dirichlet process has Dirichlet marginals.

Having defined a stochastic process G, we can now turn De Finetti's theorem around and ask what distribution is induced on a sequence $(X_1, X_2, \ldots, X_N)$ if we draw these variables independently from G and then integrate out G. The answer: the Pólya urn. We say that the Dirichlet process is the De Finetti mixing distribution underlying the Pólya urn.

In the remainder of this chapter, we denote the stochastic process defined by Eq. (5) as follows:

$$G \sim \mathcal{DP}(\alpha_0, G_0). \tag{7}$$

The Dirichlet process has two parameters, a *concentration parameter* α_0, which is proportional to the probability of obtaining a new color in the Pólya urn, and the *base measure* G_0, which is the source of the "atoms" ϕ_k.

The set of ideas introduced in this section emerged slowly over several decades. The basic definition of the Dirichlet process as a stochastic process is due to Ferguson [1973], based on earlier work by Freedman [1963]. The fact that the Dirichlet process is the De Finetti mixing distribution underlying the Pólya urn model is due to Blackwell and MacQueen [1973]. The stick-breaking construction of the Dirichlet process was presented by Sethuraman [1994]. The application of these ideas to Bayesian modeling and inference required some additional work as described in the following section.

The Dirichlet process and the stick-breaking process are essential tools in Bayesian nonparametrics. It is as important for a Bayesian nonparametrician to master them as it is for a graphical modeler to master Pearl's book. See Hjort et al. [2010] for a book-length treatment of the Dirichlet process and related ideas.

4 Dirichlet process mixtures

With an interesting class of stochastic process priors in hand, let us now describe an application of these priors to a Bayesian nonparametric modeling problem. In particular, as alluded to in the previous section, the Dirichlet process defines a prior on partitions of objects, and this prior can be used to develop a Bayesian nonparametric approach to clustering. A notable aspect of this approach is that one does not have to fix the number of clusters a priori.

Let $(X_1, X_2, \ldots, X_N)$ be a sequence of random vectors, whose realizations we want to model in terms of an underlying set of clusters. We treat these variables as exchangeable (i.e., as embedded in an infinitely-exchangeable sequence) and, as suggested by De Finetti's theorem, treat these variables as conditionally independent given an underlying random element G. In particular, letting G be a draw from a Dirichlet process, we define a *Dirichlet process mixture model* (DP-MM) [Antoniak 1974; Lo 1984] as follows:

$$\begin{aligned} G &\sim \mathcal{DP}(\alpha_0, G_0) \\ \theta_i \,|\, G &\sim G, \qquad i = 1, \ldots, N \\ x_i \,|\, \theta_i &\sim p(x_i \,|\, \theta_i), \qquad i = 1, \ldots, N, \end{aligned}$$

where $p(x_i \,|\, \theta_i)$ is a cluster-specific distribution (e.g., a Gaussian distribution, where θ_i is a mean vector and covariance matrix). This probabilistic specification is indeed directly related to De Finetti's theorem—the use of the intermediate variable θ_i is simply an expanded way to write the factor $p(x_i \,|\, G)$ in Eq. (1). In particular, G is a sum across atoms, and thus θ_i is simply one of the atoms in G, chosen with probability equal to the weight associated with that atom.

We provide a graphical model representation of the DP-MM in Figure 1. As this figure suggests, it is entirely possible to use the graphical model formalism to display Bayesian nonparametric models. Nodes in such a graph are associated with general random elements, and the distributions on these random elements can be general stochastic processes. By going to stochastic process priors we have not strayed beyond probability theory, and all of the conditional independence semantics of graphical models continue to apply.

5 Inference for Dirichlet process mixtures

Inference with stochastic processes is an entire topic of its own, and we limit ourselves here to a brief description of one particular Markov chain Monte Carlo (MCMC) inference procedure for the DP-MM. This particular procedure is due to Escobar [1994], and its virtue is simplicity of exposition, but it should not be viewed as the state of the art. See Neal [2000] for a discussion of a variety of other MCMC inference procedures for DP-MMs.

We begin by noting that the specification in Eq. (8) induces a Pólya urn marginal distribution on $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$. The joint distribution of θ and $X = (X_1, X_2, \ldots, X_N)$

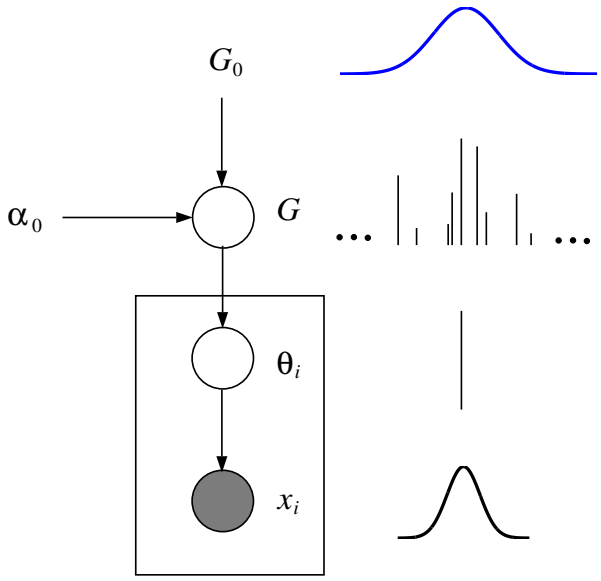


Figure 1. A graphical model representation of the Dirichlet process mixture model. Recall that the plate representation means that the parameters θ_i are drawn independently conditional on G. On the right side of the figure we have depicted specific instantiations of the random elements G and θ_i and the distribution of the observation x_i.

is thus the following product:

$$p(\theta, x) = p(\theta_1, \theta_2, \ldots, \theta_N) \prod_{i=1}^{N} p(x_i \,|\, \theta_i), \tag{8}$$

where the first factor is the Pólya urn model. This can be viewed as a product of a prior (the first factor) and a likelihood (the remaining factors).

The variable x is held fixed in inference (it is the observed data) and the goal is to sample θ. We develop a Gibbs sampler for this purpose. The main problem is to sample a particular component θ_i while holding all of the other components fixed. It is here that the property of exchangeability is essential. Because the joint probability of $(\theta_1, \ldots, \theta_N)$ is invariant to permutation, we can permute the vector to move θ_i to the end of the list. But the prior probability of the last component given all of the preceding variables is given by the urn model specification in Eq. (2). We multiply each of the distributions in this expression by the likelihood $p(x_i \,|\, \theta)$ and integrate with respect to θ. (We are assuming that G_0 and the likelihood are conjugate that this integral can be done in closed form.) The result is the conditional distribution of θ_i given the other components and given x_i. This conditional is sampled to yield the updated value of θ_i. This is done for all of the indices $i \in \{1, \ldots, N\}$ and the

process iterates.

This link between exchangeability and an efficient inference algorithm is an important one. In other more complex Bayesian nonparametric models, while we may no longer assume exchangeability, we generally aim to maintain some weaker notion (e.g., partial exchangeability) so as to have some hope of tractable inference.

6 Hierarchical Dirichlet processes

The spirit of the graphical model formalism—in particular the Bayesian network formalism based on directed graphs—is that of hierarchical Bayesian modeling. In a hierarchical Bayesian model, the joint distribution of all of the variables in the model is obtained as a product over conditional distributions, where each conditional may depend on other variables in the model. While the graphical model literature has focused almost exclusively on parametric hierarchies—where each of the conditionals is a finite-dimensional distribution—it is also possible to build hierarchies in which the components are stochastic processes. In this section we consider how to do this for the Dirichlet process.

One of the simplest and most useful ways in which hierarchies arise in Bayesian models is in the form of a conditional independence motif in which a set of variables, $(\theta_1, \theta_2, \ldots, \theta_m)$, are coupled via an underlying variable θ_0. For example, θ_i might be a Gaussian variable whose mean is equal to θ_0, which is also Gaussian; moreover, the θ_i are conditionally independent given θ_0. The inferential effect of this construction is to "shrink" the posterior distributions of θ_i towards each other. This is often a desirable effect, particularly when m is large relative to the number of observed data points.

The same tying of distributions can be done with Dirichlet processes. Recall that a Dirichlet process, $G_i \sim \mathcal{DP}(\alpha_0, G_0)$, is a random measure G_i that has a "parameter" G_0 that is itself a measure. If we treat G_0 as itself a draw from a Dirichlet process, and let the measures $\{G_1, G_2, \ldots, G_m\}$ be conditionally independent given G_0, we obtain the following hierarchy:

$$\begin{aligned} G_0 \,|\, \gamma, H &\sim \mathcal{DP}(\gamma, H) \\ G_i \,|\, \alpha, G_0 &\sim \mathcal{DP}(\alpha_0, G_0) \qquad i = 1, \ldots, m, \end{aligned}$$

where γ and H are concentration and base measure parameters at the top of the hierarchy. This construction—which is known as a *hierarchical Dirichlet process* (HDP)—yields an interesting kind of "shrinkage." Recall that G_0 is a discrete random measure, with its support on a countably infinite set of atoms. Drawing $G_i \sim \mathcal{DP}(\alpha_0, G_0)$ means that G_i will also have its support on the same set of atoms, and this will be true for each of $\{G_1, G_2, \ldots, G_m\}$. Thus these measures will share atoms. They will differ in the weights assigned to these atoms. The weights are obtained via conditionally independent stick-breaking processes.

One application of this sharing of atoms is to share mixture components across multiple clustering problems. Consider in particular a problem in which we have

m groups of data, $\{(x_{11}, x_{12}, \ldots, x_{1N_1}), \ldots, (x_{m1}, x_{m2}, \ldots x_{mN_m})\}$, where we wish to cluster the points $\{x_{ij}\}$ in the ith group. Suppose, moreover, that we view the groups as related, and we think that clusters discovered in one group might also be useful in other groups. To achieve this, we define the following *hierarchical Dirichlet process mixture model* (HDP-MM):

$$\begin{aligned} G_0 \,|\, \gamma, H &\sim \mathcal{DP}(\gamma, H) \\ G_i \,|\, \alpha, G_0 &\sim \mathcal{DP}(\alpha_0, G_0) \qquad i = 1, \ldots, m, \\ \theta_{ij} \,|\, G_i &\sim G_i \qquad j = 1, \ldots, N_i, \\ x_{ij} \,|\, \theta_{ij} &\sim F(x_{ij}, \theta_{ij}) \qquad j = 1, \ldots, N_i. \end{aligned}$$

This model is shown in graphical form in Figure 2. To see how the model achieves our goal of sharing clusters across groups, recall that the Dirichlet process clusters points within a single group by assigning the same parameter vector to those points. That is, if $\theta_{ij} = \theta_{ij'}$, the points x_{ij} and $x_{ij'}$ are viewed as belonging to the same cluster. This equality of parameter vectors is possible because both θ_{ij} and $\theta_{ij'}$ are drawn from G_i, and G_i is a discrete measure. Now if G_i and $G_{i'}$ share atoms, as they do in the HDP-MM, then points in different groups can be assigned to the same cluster. Thus we can share clusters across groups.

The HDP was introduced by Teh, Jordan, Beal and Blei [2006] and it has since appeared as a building block in a variety of applications. One application is to the class of models known as *grade of membership models* [Erosheva 2003], an instance of which is the *latent Dirichlet allocation* (LDA) model [Blei, Ng, and Jordan 2003]. In these models, each entity is associated not with a single cluster but with a set of clusters (in LDA terminology, each "document" is associated with a set of "topics"). To obtain a Bayesian nonparametric version of these models, the DP does not suffice; rather, the HDP is required. In particular, the topics for the ith document are drawn from a random measure G_i, and the random measures G_i are drawn from a DP with a random base measure G_0; this allows the same topics to appear in multiple documents.

Another application is to the hidden Markov model (HMM) where the number of states is unknown a priori. At the core of the HMM is the transition matrix, each row of which contains the conditional probabilities of transitioning to the "next state" given the "current state." Viewing states as clusters, we obtain a set of clustering problems, one for each row of the transition matrix. Using a DP for each row, we obtain a model in which the number of next states is open-ended. Using an HDP to couple these DPs, the same pool of next states is available from each of the current states. The resulting model is known as the *HDP-HMM* [Teh, Jordan, Beal, and Blei 2006]. Marginalizing out the HDP component of this model yields an urn model that is known as the *infinite HMM* [Beal, Ghahramani, and Rasmussen 2002].

Similarly, it is also possible to use the HDP to define an architecture known as

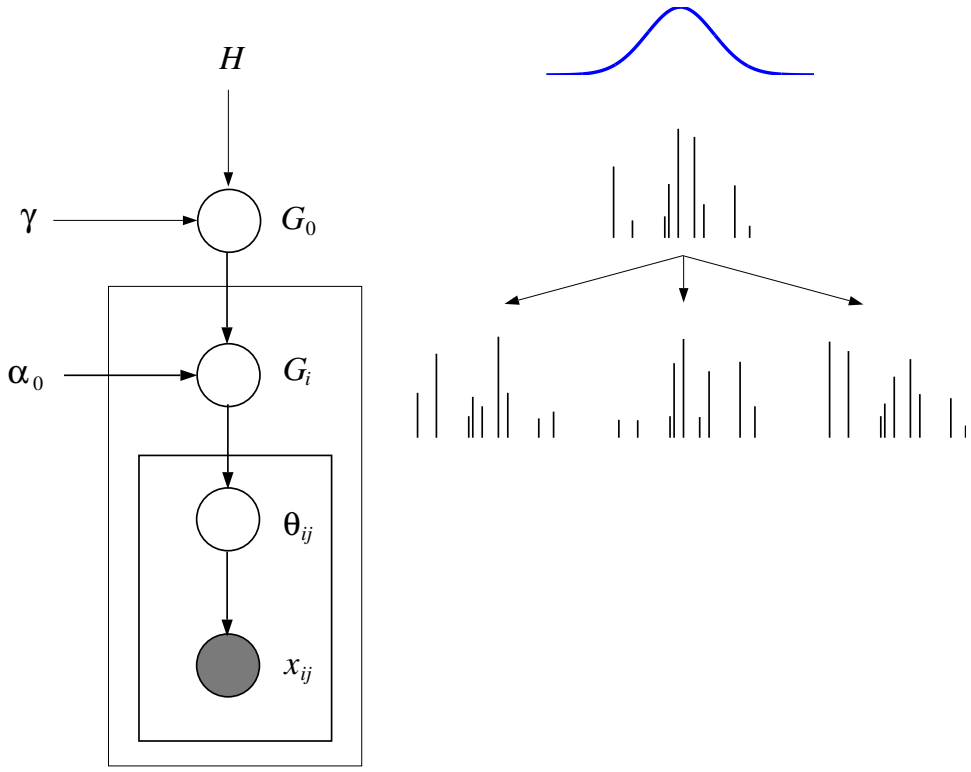


Figure 2. A graphical model representation of the hierarchical Dirichlet process mixture model. The nested plate representation means that G_0 is first drawn and held fixed, then the random measures $\{G_i\}$ are drawn independently (conditional on G_0), and finally the parameters $\{\theta_{ij}\}$ are drawn independently (conditional on G_i). On the right side of the figure we have depicted draws from G_0 and the $\{G_i\}$. Note that the atoms in these measures are at the same locations; only the weights associated with the atoms differ.

the *HDP hidden Markov tree* (HDP-HMT), a Markovian tree in which the number of states at each node in the tree is unknown a priori and the state space is shared across the nodes. The HDP-HMT has been shown to be useful in image denoising and scene recognition problems [Kivinen, Sudderth, and Jordan 2007].

Let us also mention that the HDP can be also used to develop a Bayesian nonparametric approach to probabilistic context free grammars. In particular, the HDP-PCFG of Liang, Jordan and Klein [2010] involves an HDP-based lexicalized grammar in which the number of nonterminal symbols is open-ended and inferred from data (see also Finkel, Grenager and Manning [2007] and Johnson, Griffiths and Goldwater [2007]). When a new nonterminal symbol is created at some location in a parse tree, the tying achieved by the HDP makes this symbol available at other locations in the parse tree.

There are other ways to connect multiple Dirichlet processes. One broadly useful idea is to use a Dirichlet process to define a distribution on Dirichlet processes. In particular, let $\{G_1^*, G_2^*, \ldots\}$ be independent draws from a Dirichlet process, $\mathcal{DP}(\gamma, H)$, and then let G be equal to G_k^* with probability π_k, where the weights $\{\pi_k\}$ are drawn from the stick-breaking process in Eq. (4). This construction (which can be extended to multiple levels) is known as a *nested Dirichlet process* [Rodríguez, Dunson, and Gelfand 2008]. Marginalizing over the Dirichlet processes the resulting urn model is known as the *nested Chinese restaurant process* [Blei, Griffiths, and Jordan 2010], which is a model that can be viewed as a tree of Chinese restaurants. A customer enters the tree at a root Chinese restaurant and sits at a table. This points to another Chinese restaurant, where the customer goes to dine on the following evening. The construction then recurses. Thus a given customer follows a path through the tree of restaurants, and successive customers tend to follow the same paths, eventually branching off.

These nested constructions differ from the HDP in that they do not share atoms among the multiple instances of lower-level DPs. That is, the draws $\{G_1^*, G_2^*, \ldots\}$ involve disjoint sets of atoms. The higher-level DP involves a choice among these disjoint sets.

A general discussion of some of these constructions involving multiple DPs and their relationships to directed graphical model representations can be found in Welling, Porteous and Bart [2008]. Finally, let us mention the work of MacEachern [1999], whose *dependent Dirichlet processes* provide a general formalism for expressing probabilistic dependencies among both the stick-breaking weights and the atom locations in the stick-breaking representation of the Dirichlet process.

7 Completely random measures

The Dirichlet process is not the only tool in the Bayesian nonparametric toolbox. In this section we briefly consider another class of stochastic processes that significantly expands the range of models that can be considered.

From the graphical model literature we learn that probabilistic independence of

random variables has desirable representational and computational consequences. In the Bayesian nonparametric setting, random variables arise by evaluating a random measure G on subsets of a measurable space Ω; in particular, for fixed subsets A_1 and A_2, $G(A_1)$ and $G(A_2)$ are random variables. If A_1 and A_2 are disjoint it seems reasonable to ask that $G(A_1)$ and $G(A_2)$ be independent. Such an independence relation would suggest a divide-and-conquer approach to inference.

The class of stochastic processes known as *completely random measures* are characterized by this kind of independence—for a completely random measure the random masses assigned to disjoint subsets of the sample space Ω are independent [Kingman 1967]. Note that the Dirichlet process is *not* a completely random measure—the fact that the total mass is one couples the random variables $\{G(A_i)\}$.

The Dirichlet process provides a latent representation for a clustering problem, where each entity is assigned to one and only cluster. This couples the cluster assignments and suggests (correctly) that the underlying stochastic process is not completely random. If, on the other hand, we consider a latent trait model—one in which entities are described via a set of non-mutually-exclusive binary traits—it is natural to consider completely random processes as latent representations. In particular, the *beta process* is a completely random measure in which a draw consists of a countably infinite collection of atoms, each associated with a probability, where these probabilities are independent [Hjort 1990; Thibaux and Jordan 2007]. In effect, a draw from a beta process yields an infinite collection of independent coins. Tossing these coins once yields a binary featural representation for a single entity. Tossing the coins multiple times yields an exchangeable featural representation for a set of entities.

The beta process arises via the following general construction. Consider the product space $\Omega \otimes (0,1)$. Place a product measure on this space, where the measure associated with Ω is the *base measure* B_0, and the measure associated with $(0,1)$ is obtained from the improper beta density, $cp^{-1}(1-p)^{c-1}$, where $c > 0$ is a parameter. Treating this product measure as a rate measure for a nonhomogeneous Poisson process, draw a set of points $\{(\omega_i, p_i)\}$ in the product space $\Omega \otimes (0,1)$. From these points, form a random measure on Ω as follows:

$$B = \sum_{i=1}^{\infty} p_i \delta_{\omega_i}. \tag{9}$$

The fact that we obtain an infinite collection of atoms is due to the fact that we have used a beta density that integrates to infinity. This construction is depicted graphically in Figure 3.

If we replace the beta density in this construction with other densities (generally defined on the positive real line rather than the unit interval (0,1)), we obtain other completely random measures. In particular, we obtain the *gamma process* by using an improper gamma density in place of the beta density. The gamma process provides a natural latent representation for models in which entities are

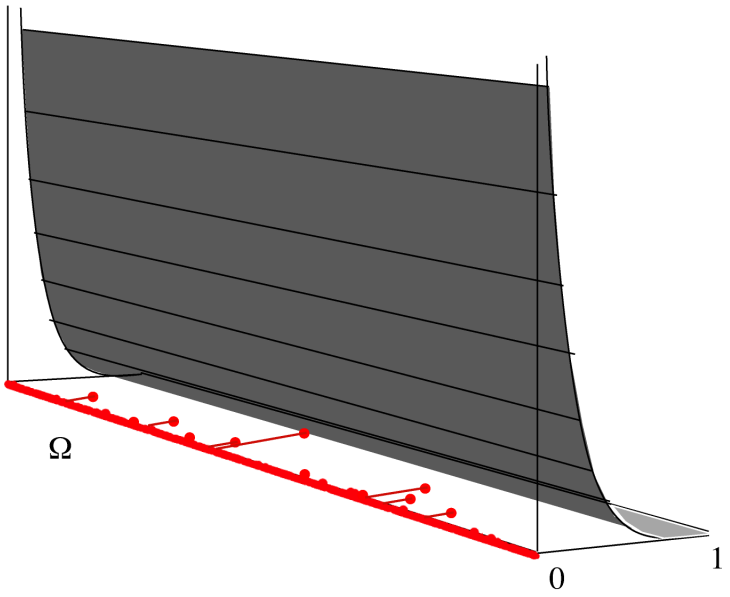


Figure 3. The construction of the beta process from a Poisson process. In this example, Ω is a bounded interval. The rate measure for the Poisson process is the shaded surface—it is the product of a uniform distribution on Ω with an improper beta distribution on $(0, 1)$. Sampling the Poisson process yields the red points in the plane, and these points are connected by line segments to the Ω-axis interval to form the random measure $B = \sum_{i=1}^{\infty} p_i \delta_{\omega_i}$.

represented by a countably infinite set of counts or rates. It is also worth noting that the Dirichlet process can be obtained by normalizing the gamma process.

Recall from our discussion in Section 2 that the Chinese restaurant process can be obtained by integrating out the Dirichlet process in a conditional independence hierarchy. In the other direction, the Dirichlet process is the random measure that is guaranteed (by exchangeability and De Finetti's theorem) to underlie the Chinese restaurant process. Given the importance of the latter model in Bayesian nonparametric modeling and computation, it is of interest to ask if there is a corresponding probability law on binary matrices obtained by integrating out the beta process. As shown by Thibaux and Jordan [2007], the answer is yes, where the probability law is the *Indian buffet process* (IBP) of Griffiths and Ghahramani [2006].

To describe the IBP, consider an Indian buffet with a countably infinite number of dishes. Let N customers arrive in sequence in the buffet line. Let Z denote

a binary-valued matrix in which the rows are customers and the columns are the dishes, and where $Z_{n,k} = 1$ if customer n samples dish k. Customer n samples dish k with probability m_k/n, where m_k is the number of customers who have previously sampled dish k; that is, $Z_{n,k} \sim \text{Ber}(m_k/n)$. (Note that this rule can be interpreted in terms of classical Bayesian analysis as sampling the predictive distribution obtained from a sequence of Bernoulli draws based on an improper beta prior.) Having sampled from the dishes previously sampled by other customers, customer n then goes on to sample an additional number of new dishes determined by a draw from a Poiss(α/n) distribution.

The connection to the beta process delineated by Thibaux and Jordan [2007] is as follows (see Teh and Jordan [2010] for an expanded discussion). Dishes in the IBP correspond to atoms in the beta process, and the independent beta/Bernoulli updating of the dish probabilities in the IBP reflects the independent nature of the atoms in the beta process. Moreover, the fact that a Poisson distribution is adopted for the number of dishes in the IBP reflects the fact that the beta process is defined in terms of an underlying Poisson process. The exchangeability of the IBP (which requires considering equivalence classes of matrices if argued directly on the IBP representation) follows immediately from the beta process construction (by the conditional independence of the rows of Z given the underlying draw from the beta process).

It is also possible to define *hierarchical beta processes* for models involving multiple beta processes that are tied in some manner [Thibaux and Jordan 2007]. This is done by simply letting the base measure for the beta process itself be drawn from the beta process:

$$\begin{aligned} B_0 &\sim \text{BP}(c_0, B_{00}) \\ B &\sim \text{BP}(c, B_0), \end{aligned}$$

where $\text{BP}(c, B_0)$ denotes the beta process with concentration parameter c and base measure B_0. This construction can be used in a manner akin to the hierarchical Dirichlet process; for example, we can use it to model groups of entities that are described by sparse binary vectors, where we wish to share the sparsity pattern among groups.

8 Conclusions

Judea Pearl's work on probabilistic graphical models yielded a formalism that was significantly more expressive than existing probabilistic representations in AI, but yet retained enough mathematical structure that it was possible to design efficient computational procedures for a wide class of useful models. In this short article, we have argued that Bayesian nonparametrics provides a framework in which this agenda can be taken further. By replacing the traditional parametric prior distributions of Bayesian analysis with stochastic processes, we obtain a rich vocabulary,

encompassing probability distributions on objects such as trees of infinite depth, partitions, subsets of features, measures and functions. We also obtain natural notions of recursion. In addition to this structural expressiveness, the Bayesian nonparametric framework also permits a wide range of distributional shapes. Finally, although we have devoted little attention to computation in this article, the stochastic processes that have been used in Bayesian nonparametrics have properties (e.g., exchangeability, independence of measure on disjoint sets) that permit the design of efficient inference algorithms. Certainly the framework is rich enough to design some intractable models, but the same holds true for graphical models. The point is that the Bayesian nonparametric framework opens the door to a richer class of useful models for AI. The growing list of successful applications of Bayesian nonparametrics testifies to the practical value of the framework [Hjort, Holmes, Mueller, and Walker 2010].

A skeptical reader might question the value of Bayesian nonparametric modeling given that for any given finite data set the posterior distribution of a Bayesian nonparametric model will concentrate on a finite set of degrees of freedom, and it would be possible in principle to build a parametric model that mimics the nonparametric model on those degrees of freedom. While this skepticism should not be dismissed out of hand—and we certainly do not wish to suggest that parametric modeling should be abandoned—this skeptical argument has something of the flavor of a computer scientist arguing that data structures such as linked lists and heaps are not needed because they can always be mimicked by fixed-dimension arrays. The nonparametric approach can lead to conceptual insights that are only available at the level of an underlying stochastic process. Moreover, by embedding a model for a fixed number of data points in a sequence of models for a growing number of data points, one can often learn something about the statistical properties of the model—this is the spirit of nonparametric statistics in general. Finally, infinite limits often lead to simpler mathematical objects.

In short, we view Bayesian nonparametrics as providing an expressive, useful language for probabilistic modeling, one which follows on directly from the tradition of graphical models. We hope and expect to see Bayesian nonparametrics have as broad of an effect on AI as that of graphical models.

References

Aldous, D. (1985). Exchangeability and related topics. In *Ecole d'Eté de Probabilités de Saint-Flour XIII–1983*, pp. 1–198. Springer, Berlin.

Antoniak, C. E. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics 2*, 1152–1174.

Beal, M. J., Z. Ghahramani, and C. E. Rasmussen (2002). The infinite hidden Markov model. In *Advances in Neural Information Processing Systems*, Volume 14, Cambridge, MA. MIT Press.

Blackwell, D. and J. B. MacQueen (1973). Ferguson distributions via Pólya urn schemes. *Annals of Statistics 1*, 353–355.

Blei, D. M., T. L. Griffiths, and M. I. Jordan (2010). The nested Chinese restaurant process and Bayesian inference of topic hierarchies. *Journal of the ACM 57*.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research 3*, 993–1022.

Erosheva, E. A. (2003). Bayesian estimation of the grade of membership model. In *Bayesian Statistics*, Volume 7, Oxford, UK, pp. 501–510. Oxford University Press.

Escobar, M. D. (1994). Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association 89*, 268–277.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics 1*, 209–230.

Finkel, J. R., T. Grenager, and C. D. Manning (2007). The infinite tree. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Freedman, D. (1963). On the asymptotic behavior of Bayes estimates in the discrete case. *Annals of Mathematical Statistics 34*, 1386–1403.

Griffiths, T. L. and Z. Ghahramani (2006). Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems*, Volume 18, Cambridge, MA. MIT Press.

Hjort, N., C. Holmes, P. Mueller, and S. Walker (2010). *Bayesian Nonparametrics: Principles and Practice*. Cambridge, UK: Cambridge University Press.

Hjort, N. L. (1990). Nonparametric Bayes estimators based on beta processes in models for life history data. *Annals of Statistics 18*, 1259–1294.

Johnson, M., T. L. Griffiths, and S. Goldwater (2007). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems*, Volume 19, Cambridge, MA. MIT Press.

Karlin, S. and H. M. Taylor (1975). *A First Course in Stochastic Processes*. New York, NY: Springer.

Kingman, J. F. C. (1967). Completely random measures. *Pacific Journal of Mathematics 21*, 59–78.

Kivinen, J., E. Sudderth, and M. I. Jordan (2007). Learning multiscale representations of natural scenes using Dirichlet processes. In *IEEE International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil.

Liang, P., M. I. Jordan, and D. Klein (2010). Probabilistic grammars and hierarchical Dirichlet processes. In *The Handbook of Applied Bayesian Analysis*, Oxford, UK. Oxford University Press.

Lo, A. (1984). On a class of Bayesian nonparametric estimates: I. Density estimates. *Annals of Statistics 12*, 351–357.

MacEachern, S. (1999). Dependent nonparametric processes. In *Proceedings of the Section on Bayesian Statistical Science*. American Statistical Association.

Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics 9*, 249–265.

Pitman, J. (2002). Combinatorial stochastic processes. Technical Report 621, Department of Statistics, University of California at Berkeley.

Rodríguez, A., D. B. Dunson, and A. E. Gelfand (2008). The nested Dirichlet process. *Journal of the American Statistical Association 103*, 1131–1154.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica 4*, 639–650.

Teh, Y. W. and M. I. Jordan (2010). Hierarchical Bayesian nonparametric models with applications. In *Bayesian Nonparametrics: Principles and Practice*. Cambridge, UK: Cambridge University Press.

Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association 101*, 1566–1581.

Thibaux, R. and M. I. Jordan (2007). Hierarchical beta processes and the Indian buffet process. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, Volume 11, San Juan, Puerto Rico.

Welling, M., I. Porteous, and E. Bart (2008). Infinite state Bayesian networks for structured domains. In *Advances in Neural Information Processing Systems*, Volume 20, Cambridge, MA. MIT Press.

11

Judea Pearl and Graphical Models for Economics

Michael Kearns

Judea Pearl's tremendous influence on the fields of artificial intelligence and machine learning began with the fundamental insight that much of traditional statistical modeling lacked expressive means for articulating known or learned structure and relationships between probabilistic entities. Judea and his early colleagues focused their efforts on a type of structure that proved to be particularly important — namely network structure, or the graph-theoretic structure that arises from pairwise influences between random variables. Judea's legacy includes not only the introduction of Bayesian networks — perhaps the most important class of probabilistic graphical models — but a rich series of results establishing firm semantics for inference, independence and causality, and efficient algorithms and heuristics for fundamental probabilistic computations. His body of work is one of those rare instances in which the contributions range from the most conceptual and philosophical to the eminently practical.

Inspired by the program established by Judea for statistical models, about a decade ago a number of us became intrigued by the possibility of replicating it in the domains of strategic, economic and game-theoretic modeling. At its highest level, the proposed metaphor was both simple and natural. Rather than a large number of random variables related by a joint distribution, imagine we have a large number of players in a (normal-form) game. Instead of the edges of a network representing direct probabilistic influences between random variables, they represent direct influences on payoffs by the actions of neighboring players. As opposed to being concerned primarily with conditional inferences on the joint distribution, we are interested in the computation of Nash and other types of equilibrium for the game. As with probabilistic graphical models, although the network succinctly articulates only local influences, in the game-theoretic setting, at equilibrium there are certainly global influences and coordination via the propagation of local effects. And finally, if we were lucky, we might hope to capture for game theory some of the algorithmic benefits that models like Bayesian networks brought to statistical modeling.

The early work following this metaphor was broadly successful in its goals. The first models proposed, which included graphical games [Kearns, Littman, and Singh 2001; Kearns 2007] and Multi-Agent Influence Diagrams [Koller and Milch 2003; Vickrey and Koller 2002], provided succinct languages for expressing strategic struc-

ture in the form of networks over the players. The NashProp algorithm for computing (approximate) Nash equilibria in graphical games was the strategic analogue of the belief propagation algorithm developed by Judea and others, and like that algorithm it came in both provably efficient form for restricted network topologies, or in more heuristic but more general form for "loopy" or highly cyclical structures [Ortiz and Kearns 2003]. There are also works carefully relating probabilistic and game-theoretic graphical models in interesting ways, as in a result showing that the distributions forming the correlated equlibria of a graphical game can be succinctly represented by a (probabilistic) Markov network using (almost) the same underlying graph structure [Kakade, Kearns, Langford, and Ortiz 2003]. Graphical games have also played an important role in some recent complexity-theoretic work, most notably the breakthrough proof establishing that the problem of computing Nash equilibria in general games for even 2 players is PPAD-complete and thus potentially intractable [Daskalakis, Goldberg, and Papadimitriou 2006].

In short, we now have a rather rich set of network-based models for game theory, and a firm understanding of their semantic and algorithmic properties. The execution of this agenda relied on Judea's work in many places for inspiration and guidance, from the very conception of the models studied to the usage of cutset conditioning and distributed dynamic programming techniques in the development of NashProp and its variants.

Encouraged by this success, more recent works have sought to expand its scope to include more specifically economic models, developing networked variants of the classical exchange economies studied by Arrow and Debreu, Fisher, and others [Kakade, Kearns, and Ortiz 2004]. Now network structure represents permissible trading partners or relationships, and again the primary solution concept of interest is an equilibrium — but now an equilibrium in prices or exchange rates that permits self-interested traders to clear the market in all goods. While, as to be expected, there are different technical details, we can again establish the algorithmic benefits of such models in the form of a price propagation algorithm for computing an approximate equilibrium. Perhaps more interesting are examinations of how network topology and equilibrium properties interact. It is worth noting that for probabilistic graphical models such as Bayesian networks, the question of what the "typical" structure looks like is somewhat nonsensical — the reply might be that there is no "typical" structure, and topology will depend highly on the domain (whether it be machine vision, medical diagnosis, and so on). In contrast, the emerging literature on social and economic networks is indeed beginning to establish at least broad topological features that arise frequently in empirical networks. This invites, for example, results establishing that if the network structure exhibits a heavy-tailed distribution of connectivity (degrees), agent wealths at equilibrium will also be distributed in highly unequal fashion [Kakade, Kearns, Ortiz, Pemantle, and Suri 2005] . Thus social network structure may be (just one) explanation for observed disparities in wealth.

The lines of research sketched above continue to grow and deepen, and have become one of the many topics of mutual interest between computer scientists, economists and sociologists. Those of us who were exposed to and inspired by Judea's work in probabilistic graphical models were indeed most fortunate to have had the opportunity to help initiate a fundamental and interdisciplinary subject only shortly before social, economic and technological network structure became a topic of such general interest.

Thank you Judea!

References

Daskalakis, C., P. Goldberg, and C. Papadimitriou (2006). The complexity of computing a Nash equilibrium. In *Proceedings of the Thirty-Eighth ACM Symposium on the Theory of Computing*, pp. 71–78. ACM Press.

Kakade, S., M. Kearns, J. Langford, and L. Ortiz (2003). Correlated equilibria in graphical games. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pp. 42–47. ACM Press.

Kakade, S., M. Kearns, and L. Ortiz (2004). Graphical economics. In *Proceedings of the 17th Annual Conference on Learning Theory*, pp. 17–32. Springer Berlin.

Kakade, S., M. Kearns, L. Ortiz, R. Pemantle, and S. Suri (2005). Economic properties of social networks. In L. Saul, Y. Weiss, and L. Bottou (Eds.), *Advances in Neural Information Processing Systems 17*, pp. 633–640. MIT Press.

Kearns, M. (2007). Graphical games.

Kearns, M., M. Littman, and S. Singh (2001). Graphical models for game theory. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 253–260. Morgan Kaufmann.

Koller, D. and B. Milch (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior 45*(1), 181–221.

Ortiz, L. and M. Kearns (2003). Nash propagation for loopy graphical games. In S. Becker, S. Thrun, and K. Obermayer (Eds.), *Advances in Neural Information Processing Systems 15*, pp. 793–800. MIT Press.

Vickrey, D. and D. Koller (2002). Multi-agent algorithms for solving graphical games. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 345–351. AAAI Press.

12

Belief Propagation in Loopy Graphs

Daphne Koller

1 Introduction and Historical Perspective

Of Judea Pearl's many seminal contributions, perhaps the one that has had the greatest impact (so far) is the development of key ideas in the representation, semantics, and inference of *probabilistic graphical models*. This formalism provides an elegant and practical framework for representing a probability distribution over a high-dimensional space defined as the set of possible assignments to a set of *random variables* $X_1, \ldots, X_n$. The number of such assignments grows exponentially in n, but due to the key insights of Pearl and others, we now understand how *conditional independence* properties of the joint probability distribution $P(X_1, \ldots, X_n)$ allow it to be represented compactly and naturally using a graph annotated with local probabilistic interactions (see section 2). The family of probabilistic graphical models includes *Bayesian networks*, which are based on directed graphs, and *Markov networks* (also called *Markov random fields*), which use undirected graphs.

The number of applications of this framework is far too large to enumerate. One of the earliest applications is in the area of medical diagnosis. Here, we might have hundreds of random variables, representing predisposing factors, possible diseases, symptoms, and test results. The framework of Bayesian networks allows such a distribution to be encoded using a limited set of local (directed) interactions, such as those between a disease and its predisposing factors, or those between a symptom and the diseases that cause it (e.g., [Heckerman, Horvitz, and Nathwani 1992; Shwe, Middleton, Heckerman, Henrion, Horvitz, Lehmann, and Cooper 1991]). In a very different application, we might want to encode a probability distribution over possible segmentations of an image — labelings of the pixels in the image into different semantic categories (such as sky, grass, building, person, etc.). Here, we have a random variable for each pixel in the image (hundreds of thousands even for the smallest images), representing its possible labels. And yet, the distribution over the space of possible segmentations is often well-represented in a Markov network, using only terms that encode each pixel's individual preferences over possible labels and (undirected) local interactions between the labels of adjacent pixels (see Szeliski, Zabih, Scharstein, Veksler, Kolmogorov, Agarwala, Tappen, and Rother [2008] for a survey).

A key question, however, is how to use this compact representation to answer questions about the distribution. The most common types of questions are *condi-*

tional probability queries, where we wish to infer the probability distribution over some (small) subset of variables given evidence concerning some of the others; for example, in the medical diagnosis setting, we might want to infer the distribution over each possible disease given observations about the patient's predisposing factors, symptoms, and some test results. A second common type of query is the *maximum a posteriori* (or *MAP*) query, where we wish to find the most likely joint assignment to all of our random variables; for example, we often wish to find the most likely joint segmentation to all of the pixels in an image.

In general, it is not difficult to show that both of these inference problems are NP-hard [Cooper 1990; Shimony 1994], yet (as always) this is not end of the story. In their seminal paper, Kim and Pearl [1983] presented an algorithm that passes messages between the nodes in the Bayesian network graph to propagate beliefs between them. The algorithm was developed in the context of singly connected directed graphs, also known as *polytrees*, where there is at most one path (ignoring edge directionality) between each pair of nodes. In this case, the message passing process produces correct posterior beliefs for each node in the graph.

Pearl also considered what happens when the algorithm is executed (without change) over a loopy (multiply connected) graph. In his seminal book, Pearl [1988] says:

> When loops are present, the network is no longer singly connected and local propagation schemes will invariably run into trouble . . . If we ignore the existence of loops and permit the nodes to continue communicating with each other as if the network were singly connected, messages may circulate indefinitely around the loops and the process may not converge to a stable equilibrium . . . Such oscillations do not normally occur in probabilistic networks . . . which tend to bring all messages to some stable equilibrium as time goes on. However, this asymptotic equilibrium is not coherent, in the sense that it does not represent the posterior probabilities of all nodes of the networks.

As a consequence of these problems, the idea of *loopy belief propagation* was largely abandoned for many years.

The revival of this approach is surprisingly due to a seemingly unrelated advance in coding theory. The area of coding addresses the problem of sending messages over a noisy channel, and recovering it from the garbled result. We send a k-bit message, redundantly coded using n bits. These n bits are sent over the noisy channel, so the received bits are possibly corrupted. The decoding task is to recover the original message from the bits received. The *bit error rate* is the probability that a bit is ultimately decoded incorrectly. This error rate depends on the code and decoding algorithm used and on the amount of noise in the channel. The *rate* of a code is k/n — the ratio between the number of bits in the message and the number of bits used to transmit it. In 1948, Claude Shannon provided a theoretical analysis of

the coding problem [Shannon 1948]. For a given rate, Shannon provided an upper bound on the maximum noise level that can be tolerated while still achieving a certain bit error rate, no matter which code is used. Shannon also showed that there exist channel codes that achieve this limit, but his proof was nonconstructive — he did not present practical encoders and decoders that achieve this limit.

Since Shannon's landmark result, multiple codes were suggested. However, despite a gradual improvement in the quality of the code (bit-error rate for a given noise level), none of the codes even came close to the Shannon limit. The big breakthrough came in the early 1990s, when Berrou, Glavieux, and Thitimajshima [1993] came up with a new scheme that they called a *turbocode*, which, empirically, came much closer to achieving the Shannon limit than any other code proposed up to that point. However, their decoding algorithm had no theoretical justification, and, while it seemed to work well in real examples, could be made to diverge or converge to the wrong answer. The second big breakthrough was the subsequent realization, due to McEliece, MacKay, and Cheng [1998] and Frey and MacKay [1997] that the turbocoding procedure was simply performing loopy belief propagation message passing on a Bayesian network representing the probability model for the code and the channel noise!

This revelation had a tremendous impact on both the coding theory community and the graphical models community. For the former, loopy belief propagation provides a general-purpose algorithm for decoding a large family of codes. By separating the algorithmic question of decoding from the question of the code design, it allowed the development of many new coding schemes with improved properties. These codes have come much, much closer to the Shannon limit than any previous codes, and they have revolutionized both the theory and the practice of coding. For the graphical models community, it was the astounding success of loopy belief propagation for this application that led to the resurgence of interest in these approaches. Subsequent work showed that this algorithm works very well in practice on a broad range of other problems (see, for example, Weiss [1996] and Murphy, Weiss, and Jordan [1999]), leading to a large amount of work on this topic. In this short paper, we review only some of the key ideas underlying this important class of methods; see section 6 for some discussion and further references.

2 Background

2.1 Probabilistic Graphical Models

Probabilistic graphical models are a general family of representations for probability distributions over high-dimensional spaces. Specifically, our goal is to encode a joint probability distribution over the possible assignments to a set of random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$. We focus on the discrete setting, where each random variable X_i takes values in some set $Val(X_i)$. In this case, the number of possible assignments grows exponentially with the number of variables n, making an explicit enumeration of the joint distribution infeasible.

Probabilistic graphical models use a factored representation to avoid the exponential representation of the joint distribution. In the most general setting, the distribution is defined via a set of *factors* Φ. A factor ϕ_k is defined over a *scope* $Scope[\phi_k] = \boldsymbol{X}_k \subseteq \boldsymbol{X}$; the factor is a function $\phi_k : Val(\boldsymbol{X}_k) \mapsto \mathbb{R}^+$. The joint distribution P_Φ is defined by multiplying together all the factors in Φ, and renormalizing to form a distribution:

$$\begin{aligned} \tilde{P}_\Phi(\boldsymbol{x}) &= \prod_k \phi_k(\boldsymbol{x}_k) \\ Z &= \sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} \tilde{P}_\Phi(\boldsymbol{x}) \\ P_\Phi(\boldsymbol{x}) &= \frac{1}{Z}\tilde{P}_\Phi(\boldsymbol{x}). \end{aligned}$$

For example, if we have a distribution over $\{X_1, \ldots, X_3\}$, defined by two pairwise factors $\phi_1(X_1, X_2)$ and $\phi_2(X_2, X_3)$, then $\tilde{P}_\Phi(x_1, x_2, x_3) = \phi_1(x_1, x_2) \cdot \phi_2(x_2, x_3)$. The normalizing constant Z is historically called the *partition function*.

This factorization is generally tied to a graph whose nodes represent the variables $X_1, \ldots, X_n$ and whose edge structure corresponds to the factorization of the distribution. In particular, the *Markov network* representation uses an undirected graph $\mathcal{H}$ over the nodes $X_1, \ldots, X_n$. A factorized distribution P_Φ is said to *factorize* over $\mathcal{H}$ if, for every factor $\phi_k \in P_\Phi$, we have that $Scope[\phi_k]$ is a completely connected subgraph in $\mathcal{H}$ (so that every $X_i, X_j \in Scope[\phi_k]$ are connected by an undirected edge in $\mathcal{H}$). A *Bayesian network* uses a directed acyclic graph $\mathcal{G}$ to represent the distribution. In this case, each variable X_i has a set of *Parents* $\mathbf{Pa}^{\mathcal{G}}_{X_i}$. The distribution is now parameterized using a set of factors Φ which take the form $P(X_i \mid \mathbf{Pa}^{\mathcal{G}}_{X_i})$. In other words, in this factorization, we have precisely one factor for each variable X_i containing $\{X_i\} \cup \mathbf{Pa}^{\mathcal{G}}_{X_i}$, and this factor is locally normalized so that $\sum_{x_i \in Val(X_i)} P(x_i \mid \boldsymbol{u}_i) = 1$ for each assignment $\boldsymbol{u}_i \in Val(\mathbf{Pa}^{\mathcal{G}}_{X_i})$. For this set of factors, the partition function is guaranteed to be 1, and so we can now say that a distribution P *factorizes* over $\mathcal{G}$ if it can be written as:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathbf{Pa}^{\mathcal{G}}_{X_i}),$$

a formula typically known as the *chain rule for Bayesian networks*.

We note that the graph structure associated with a distribution P_Φ reveal independence properties that hold in P_Φ. That is, an examination of the network structure over which P_Φ factorizes provides us with a set of independencies that are guaranteed to hold for P_Φ, regardless of the specific parameterization. The connection between the graph structure and the independencies in the distribution was a large focus of the early work on graphical models, and many of the key contributions were developed by Pearl and his students. However, this topic is outside the scope of this paper.

2.2 Inference Tasks

Our probabilistic model $P(X_1, \ldots, X_n)$ often defines a general-purpose distribution that can be applied in multiple cases. For example, in a medical diagnosis setting, we typically have a distribution over diseases, symptoms, and test results that might hold for an entire patient population. Given a particular patient, we might observe values values for some subset of the variables (say some symptoms and test results), so that we know $\boldsymbol{E} = \boldsymbol{e}$. Thus, we now have a conditional distribution $P(\boldsymbol{W} \mid \boldsymbol{E} = \boldsymbol{e})$, where $\boldsymbol{W} = \boldsymbol{X} - \boldsymbol{E}$. This conditional distribution has the form $P(\boldsymbol{W}, \boldsymbol{e})/P(\boldsymbol{e})$, where $P(\boldsymbol{e}) = \sum_{\boldsymbol{W}} P(\boldsymbol{W}, \boldsymbol{e})$ is a normalizing constant.

Importantly, if our distribution is derived as P_Φ for some set of factors Φ, we can easily obtain a factored form for the numerator by simply *reducing* each factor in Φ to contain only those entries that are consistent with $\boldsymbol{E} = \boldsymbol{e}$. The resulting reduced factors can be multiplied to produce $P(\boldsymbol{W}, \boldsymbol{e})$. If the original distribution P_Φ factorizes over a Markov network $\mathcal{H}$, the conditional distribution now factorizes over the Markov network where we simply remove the nodes in $\boldsymbol{E}$ from the graph. If the original distribution P_Φ factorizes over a Bayesian network, the resulting reduced factors no longer satisfy the local normalization requirements defined by the directed graph. Since these local normalization requirements (even if they hold) do not play a role in most inference algorithms, it is generally easier to ignore them and simply consider a distribution defined by a set of (possibly reduced) factors Φ. This will be our focus for the rest of the discussion.

In this setting, we generally consider two main inference tasks. The first is computing the marginal distribution over one or more query variables; for example, we might want to compute

$$P_\Phi(\boldsymbol{Y}) = \sum_{\boldsymbol{W}} P_\Phi(\boldsymbol{Y}, \boldsymbol{W}) = \frac{1}{Z} \sum_{\boldsymbol{W}} \prod_k \phi_k(\boldsymbol{Y}_k, \boldsymbol{W}_k),$$

where $\boldsymbol{Y}_k, \boldsymbol{W}_k$ represents the assignment in $\boldsymbol{Y}, \boldsymbol{W}$ to $\boldsymbol{X}_k = \mathit{Scope}[\phi_k]$. The form of this expression gives rise to the name *sum-product* for this type of inference task. This task is used in the many settings (such as medical diagnosis, for example) where we wish to compute the posterior distribution over some small subset of variables given our current observations.

A second task is computing a single joint assignment $\boldsymbol{x}$ to all variables $\boldsymbol{X}$ that achieves the highest joint probability:

$$\boldsymbol{x}^{map} = \text{argmax}_{\boldsymbol{x}} P_\Phi(\boldsymbol{x}) = \text{argmax}_{\boldsymbol{x}} \frac{1}{Z} \tilde{P}_\Phi(\boldsymbol{x}) = \text{argmax}_{\boldsymbol{x}} \tilde{P}_\Phi(\boldsymbol{x}), \tag{1}$$

where the partition function cancels since it has no effect on the choice of maximizing assignment. This assignment $\boldsymbol{x}^{map}$ is known as the *maximum a posteriori*, or *MAP*, assignment. The form of this expression gives rise to the name *max-product* for this inference task. MAP queries are used in tasks where we wish to find a single consistent joint hypothesis about the unobserved variables in our domain, for

example, a single consistent segmentation of an image or the most likely utterance in a speech recognition system.

3 Exact Inference: Clique Trees

One approach to addressing the problem of exact inference in a graphical model is by using a graphical structure called a *clique tree*. Let $\mathcal{T}$ be an undirected graph, each of whose nodes i is associated with a subset $\boldsymbol{C}_i \subseteq \mathcal{X}$. We say that $\mathcal{T}$ is *family-preserving* with respect to Φ if each factor $\phi \in \Phi$ must be associated with a cluster $\boldsymbol{C}$, denoted $\alpha(\phi)$, such that $Scope[\phi] \subseteq \boldsymbol{C}_i$. Each edge between a pair of clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ is associated with a *sepset* $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$. We say that $\mathcal{T}$ satisfies the *running intersection property* if, whenever there is a variable X such that $X \in \boldsymbol{C}_i$ and $X \in \boldsymbol{C}_j$, then for every edge e in the unique path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, we have that $X \in \boldsymbol{S}_e$. If $\mathcal{T}$ satisfies the family-preservation and running-intersection properties, we say that it is a *clique tree* for the graphical model defined by Φ.

We can now specify a general inference algorithm that can be implemented via *message passing* in a clique tree. Let $\mathcal{T}$ be a clique tree with the cliques $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$. Roughly speaking, we begin by multiplying the factors assigned to each clique, resulting in our initial potentials. We then use the clique-tree data structure to pass messages between neighboring cliques.

More precisely, recall that each factor $\phi \in \Phi$ is assigned to some clique $\alpha(\phi)$. We define the *initial potential* of $\boldsymbol{C}_j$ to be:

$$\pi_j^0[\boldsymbol{C}_j] = \prod_{\phi \,:\, \alpha(\phi)=j} \phi.$$

Because each factor is assigned to exactly one clique, we have that

$$\prod_{\phi} \phi = \prod_{j} \pi_j^0.$$

We now use the clique tree structure to pass messages. The message from $\boldsymbol{C}_i$ to another clique $\boldsymbol{C}_j$ is computed using the following *sum-product message passing* operation:

$$\delta_{i \to j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \pi_i^0 \times \prod_{k \in (\mathcal{N}_i - \{j\})} \delta_{k \to i}. \tag{2}$$

In words, the clique $\boldsymbol{C}_i$ multiplies all incoming messages from its other neighbors with its initial clique potential, resulting in a factor ψ whose scope is the clique. It then sums out all variables except those in the sepset between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, and sends the resulting factor as a message to $\boldsymbol{C}_j$.

This computation can be scheduled in a variety of ways. Most generally, we say that $\boldsymbol{C}_i$ is ready to transmit a message to $\boldsymbol{C}_j$ when $\boldsymbol{C}_i$ has messages from all of its neighbors except from $\boldsymbol{C}_j$. In such a setting, $\boldsymbol{C}_i$ can compute the message $\delta_{i \to j}(\boldsymbol{S}_{i,j})$ by multiplying its initial potential with all of its incoming messages except the one from $\boldsymbol{C}_j$, and then eliminate the variables in $\boldsymbol{C}_i - \boldsymbol{S}_{i,j}$. Although the algorithm is

defined asynchronously, the message-passing process performed by the algorithm is equivalent to a much more systematic process that consists of an *upward pass* where all messages are sent toward a clique known as the *root*, and then a *downward pass* where messages are sent to all the leaves.

At the end of this process, all cliques have all of their incoming messages, at which point each clique can compute a factor called the *beliefs*:

$$\pi_r[\boldsymbol{C}_i] = \pi_i^0 \times \prod_{k \in \mathcal{N}_{\boldsymbol{C}_i}} \delta_{k \to i}.$$

This algorithm, when applied to a clique tree that satisfies the family preservation and running intersection property, computes messages and beliefs repesenting well-defined expressions. In particular, we can show that the message passed from $\boldsymbol{C}_i$ to $\boldsymbol{C}_j$ is the product of all the factors in $\mathcal{F}_{\prec(i \to j)}$, marginalized over the variables in the sepset (that is, summing out all the others):

$$\delta_{i \to j}(\boldsymbol{S}_{i,j}) = \sum_{\mathcal{V}_{\prec(i \to j)}} \prod_{\phi \in \mathcal{F}_{\prec(i \to j)}} \phi.$$

It then follows that, when the algorithm terminates, we have, for each clique i

$$\pi_i[\boldsymbol{C}_i] = \sum_{\mathcal{X} - \boldsymbol{C}_i} \tilde{P}_\Phi(\mathcal{X}), \tag{3}$$

that is, the value of the unnormalized measure $\tilde{P}_\Phi$, marginalized over the variables in $\boldsymbol{C}_i$.

We note that this expression holds for all cliques; thus, in one upward-downward pass of the algorithm, we obtain all of the marginals of all of the cliques in the network, from which we can also obtain the marginals over all variables: to compute the marginal probability over a particular variable X, we can select a clique whose scope contains X, and marginalize all variables other than X. This capability is very valuable in many applications; for example, in a medical-diagnosis setting, we generally want the probability of several possible diseases.

An important consequence of (3) is that we obtain the same marginal distribution over X regardless of the from which we extracted it. More generally, for any two adjacent cliques $\boldsymbol{C}_i$, we must have that

$$\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \pi_i[\boldsymbol{C}_i] = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \pi_j[\boldsymbol{C}_j].$$

In this case, we say that $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ are *calibrated*.

These message passing rules, albeit in a simplified form, were first developed in Pearl's analysis [Kim and Pearl 1983; Pearl 1988] on inference in singly connected (polytree) Bayesian networks. In this case, each clique represents a family — a set comprising an individual variable and its parents — and the connections between the cliques follow the structure of the original Bayesian network. With that mapping,

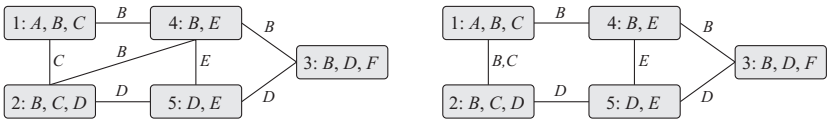


Figure 1. **Two examples of generalized cluster graph for an MRF** with potentials over $\{A, B, C\}$, $\{B, C, D\}$, $\{B, D, F\}$, $\{B, D\}$ and $\{D, E\}$.

the clique tree message passing algorithm we described is precisely Pearl's belief propagation algorithm. The more general case of this particular algorithm was developed by Shafer and Shenoy [1990], who described it in a much broader form that applies to many factored models other than probabilistic graphical models. An alternative but ultimately equivalent message passing scheme (which uses a sum-product-divide sequence for each message passing step) was was developed in parallel, in a series of papers by Lauritzen and Spiegelhalter [1988] and Jensen, Olesen, and Andersen [1990].

4 Belief Propagation in Loopy Graphs

While very compelling, the clique tree algorithm often hits against significant computational barriers. There are many graphical models for which any legal clique tree — one that satisfies family preservation and running intersection — has cliques that are very large. For example, any clique tree for a pairwise Markov network encoding an $n \times n$ grid (a class of network commonly used in computer vision applications) has cliques involving at least n variables. In such cases, inference in a clique tree requires computation that is exponential in the size of the graphical model. In a sense, this is inevitable in the worst case, given that the exact inference problem is NP-hard. However, since this exponential blowup arises in many applications of significant practical impact, another solution is necessary.

4.1 Cluster Graphs

One generalization of the basic algorithm relaxes the requirements on the message passing structure. In particular, we generalize the clique tree structure to that of a *cluster graph*. This structure is also comprised of a set of clusters $\boldsymbol{C}_i \subseteq \mathcal{X}$ connected by edges. There are three important differences: (1) a cluster graph need not be a tree; (2) the sepsets are required only to satisfy $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$; and (3) we have a modified version of the running intersection property, where we require that whenever $X \in \boldsymbol{C}_i$ and $X \in \boldsymbol{C}_j$, there is exactly one path between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ for which $X \in \boldsymbol{S}_e$ for all edges e in the path. The generalized running intersection property implies that all edges associated with X form a tree that spans all the clusters that contain X. Thus, intuitively, there is only a single path by which information *that is directly about* X can flow in the graph. Both parts of this assumption are significant. The fact that some path must exist forces information

about X to flow between all clusters that contain it, so that, in a calibrated cluster graph, all clusters must agree about the marginal distribution of X. The fact that there is at most one path prevents loops in the cluster graph where all of the clusters contain X. In graphs that contain such loops, a message passing algorithm can propagate information about X endlessly around the loop, making the beliefs more extreme due to "cyclic arguments."

Importantly, however, since the graph is not necessarily a tree, the same pair of clusters might also be connected by other paths. For example, in the cluster graph of figure 1a, we see that the edges labeled with B form a subtree that spans all the clusters that contain B. However, there are loops in the graph. For example, there are two paths from $\boldsymbol{C}_3 = \{B, D, F\}$ to $\boldsymbol{C}_2 = \{B, C, D\}$. The first, through $\boldsymbol{C}_4$, propagates information about B, and the second, through $\boldsymbol{C}_5$, propagates information about D. Thus, we can still get circular reasoning, albeit less directly than we would in a graph that did not satisfy the running intersection property. Note that while in the case of trees the definition of running intersection implied that $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$, in a graph this equality is no longer enforced by the running intersection property. For example, cliques $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ in figure 1a have B in common, but $\boldsymbol{S}_{1,2} = \{C\}$.

We note that there are many possible choices for the cluster graph, and the decision on which to use can make a significant difference to the algorithm. In particular, different graphs can lead to very different computational cost, different convergence behavior and even different answers.

EXAMPLE 1. Consider, for example, the cluster graphs $\mathcal{U}_1$ and $\mathcal{U}_2$ of figure 1a and figure 1b. Both are fairly similar, yet in $\mathcal{U}_2$ the edge between $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ involves the marginal distribution over B and C. On the other hand, in $\mathcal{U}_1$, we propagate the marginal only over C. Intuitively, we expect inference in $\mathcal{U}_2$ to better capture the dependencies between B and C. For example, assume that the potential of $\boldsymbol{C}_1$ introduces strong correlations between B and C (say $B = C$). In $\mathcal{U}_2$, this correlation is conveyed to $\boldsymbol{C}_2$ directly. In $\mathcal{U}_1$, the marginal on C is conveyed on the edge (1–2), while the marginal on B is conveyed through $\boldsymbol{C}_4$. In this case, the strong dependency between the two variables is lost. In particular, if the marginal on C is diffuse (close to uniform), then the message $\boldsymbol{C}_1$ sends to $\boldsymbol{C}_4$ will also have a uniform distribution on B, and from $\boldsymbol{C}_2$'s perspective the messages on B and C will appear as two independent variables.

One class of networks for which a simple cluster graph construction exists is the class of *pairwise Markov networks*. In these networks, we have a univariate potential $\phi_i[X_i]$ over each variable X_i, and in addition a pairwise potential $\phi_{(i,j)}[X_i, X_j]$ over some pairs of variables. These pairwise potentials correspond to edges in the Markov network. Many problems are naturally formulated as pairwise Markov networks, such as the grid networks common in computer vision applications. Indeed, if we are willing to transform our variables, any distribution can be reformulated as a

pairwise Markov network.

One straightforward transformation of a pairwise Markov network into a cluster graph is as follows: For each potential, we introduce a corresponding cluster, and put edges between the clusters that have overlapping scope. In other words, there is an edge between the cluster $\boldsymbol{C}_{(i,j)}$ that corresponds to the edge X_i—X_j and the clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ that correspond to the univariate factors over X_i and X_j. Because there is a direct correspondence between the clusters in the cluster graphs and variables or edges in the original Markov network, it is often convenient to think of the propagation steps as operations on the original network. Moreover, since each pairwise cluster has only two neighbors, we can consider two propagation steps along the path $\boldsymbol{C}_i$—$\boldsymbol{C}_{(i,j)}$—$\boldsymbol{C}_j$ as propagating information between X_i and X_j.

A highly related transformation applies to Bayesian networks. Here, as in the case of polytrees, we define a cluster $\boldsymbol{C}_i$ for each family $\{X_i\} \cup \mathbf{Pa}_{X_i}$. For every edge $X_i \rightarrow X_j$, we connect $\boldsymbol{C}_i$ to $\boldsymbol{C}_j$ via a sepset whose scope is X_i. With this cluster graph construction, the message passing algorithm described below is performing precisely the loopy belief propagation for Bayesian networks first proposed by Pearl.

A related but more general construction that applies to arbitrary sets of factors is the *Bethe cluster graph*. This construction uses a bipartite graph: The first layer consists of "large" clusters, with one cluster for each factor ϕ in Φ, whose scope is $Scope[\phi]$. These clusters ensure that we satisfy the family-preservation property. The second layer consists of "small" univariate clusters, one for each random variable. Finally, we place an edge between each univariate cluster X on the second layer and each cluster in the first layer that includes X; the scope of this edge is X itself. We can easily verify that this cluster graph is a proper one. First, by construction, it satisfies the family preservation property. Second, the edges that mention a variable X form a star-shaped subgraph with edges from the univariate cluster for X to all the large clusters that contain X. The construction of this cluster graph is simple and can easily be automated.

The broader notion of message passing on a more general cluster graph was first proposed by Yedidia, Freeman, and Weiss [2000] and Dechter, Kask, and Mateescu [2002]. Indeed, Yedidia, Freeman, and Weiss [2000, 2005] defined an even more general notion of message passing on a *region graph*, which is outside the scope of this paper.

4.2 Message Passing in Cluster Graphs

How do we perform inference in a cluster graph rather than a clique tree? From the local perspective of a single cluster $\boldsymbol{C}_i$, there is not much difference between a cluster graph and a clique tree. The cluster is related to each neighbor through an edge that conveys information on variables in the sepset. Thus, we can transmit information by simply having one cluster pass a message to the other. Of course, as the graph may have no leaves, we might initially not have any cliques that are ready to transmit. We address this issue by initializing all messages $\delta_{i \rightarrow j} = \mathbf{1}$. Clusters

Procedure CGraph-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{U}$ // Generalized cluster graph Φ
)
1 **for** each cluster $\boldsymbol{C}_i$
2 $\pi_i \leftarrow \prod_{\phi \,:\, \alpha(\phi)=i} \phi$
3 **for** each edge $(i–j) \in \mathcal{E}_{\mathcal{U}}$
4 $\delta_{i \to j} \leftarrow 1$; $\delta_{j \to i} \leftarrow 1$
5
6 **while** graph is not calibrated
7 Select $(i–j) \in \mathcal{E}_{\mathcal{U}}$
8 $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \left(\pi_i^0 \times \prod_{k \in (\mathcal{N}_i - \{j\})} \delta_{k \to i} \right)$
9
10 **for** each clique i
11 $\pi_i \leftarrow \pi_i^0 \times \prod_{k \in \mathcal{N}_i} \delta_{k \to i}$
12 **return** $\{\pi_i\}$

Figure 2. Calibration using sum-product belief propagation in a cluster graph

then pass messages to their neighbors, summarizing the current beliefs derived from their own initial potentials and from the messages received by their neighbors. The algorithm is shown in figure 2. Convergence is achieved when the cluster graph is *calibrated*; that is, if for each edge $(i–j)$, connecting the clusters $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, we have that

$$\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \pi_i = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \pi_j.$$

Note that this definition is weaker than cluster tree calibration, since the clusters do not necessarily agree on the joint marginal of all the variables they have in common, but only on those variables in the sepset. However, if a calibrated cluster graph satisfies the running intersection property, then the marginal of a variable X is identical in all the clusters that contain it. This algorithm clearly generalizes the clique-tree message-passing algorithm described earlier.

EXAMPLE 2. With this framework in hand, we can now revisit the message decoding task. Assume that we wish to send a k-bit message $u_1, \ldots, u_k$. We *code* the message using a number of bits $x_1, \ldots, x_n$, which are then sent over the noisy channel, resulting in a set of (possibly corrupted) outputs $y_1, \ldots, y_n$. The *message decoding* task is to recover an estimate $\hat{u_1}, \ldots, \hat{u_k}$ from $y_1, \ldots, y_n$. We first observe that message decoding can easily be reformulated as a probabilistic inference task: We have a prior over the message bits $\boldsymbol{U} = \langle U_1, \ldots, U_k \rangle$, a (usually deterministic) function that defines how a message is converted into a sequence of transmitted bits $X_1, \ldots, X_n$, and another (stochastic) model that defines how the channel ran-

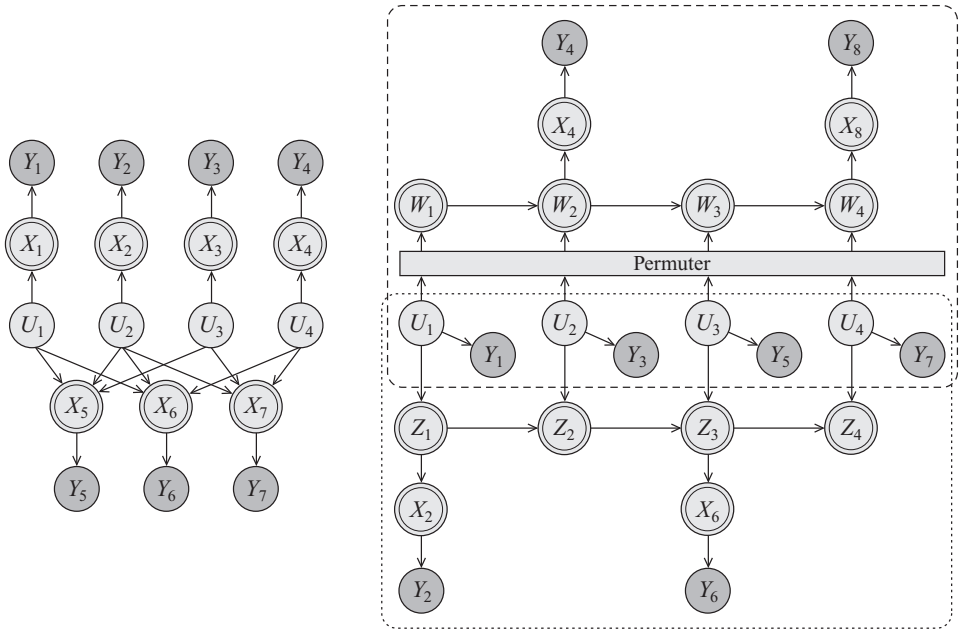


Figure 3. **Two examples of codes** (a) A $k = 4, n = 7$ parity check code, where every four message bits are sent along with three bits that encode parity checks. (b) A $k = 4, n = 8$ turbocode. Here, the $\boldsymbol{X}^a$ bits X_1, X_3, X_5, X_7 are simply the original bits U_1, U_2, U_3, U_4 and are omitted for clarity of the diagram; the $\boldsymbol{X}^b$ bits use a *shift register* — a state bit that changes with each bit of the message, where the ith state bit depends on the $(i-1)$st state bit and on the ith message bit. The code uses two shift registers, one applied to the original message bits and one to a set of permuted message bits (using some predetermined permutations). The sent bits contain both the original message bits and some number of the state bits.

domly corrupts the X_i's to produce Y_i's. The decoding task can then be viewed as finding the most likely joint assignment to $\boldsymbol{U}$ given the observed message bits $\boldsymbol{y} = \langle y_1, \ldots, y_n \rangle$, or (alternatively) as finding the posterior $P(U_i \mid \boldsymbol{y})$ for each bit U_i. The first task is a MAP inference task, and the second task one of computing posterior probabilities. Unfortunately, the probability distribution is of high dimension, and the network structure of the associated graphical model is quite densely connected and with many loops.

The turbocode approach, as first proposed, comprised both a particular coding scheme, and the use of a message passing algorithm to decode it. The coding scheme transmits two sets of bits: one set comprises the original message bits $\boldsymbol{X}^a = \langle X_1^a, \ldots, X_k^a \rangle = \boldsymbol{u}$, and the second some set $\boldsymbol{X}^b = \langle X_1^b, \ldots, X_k^b \rangle$ of transformed bits

(like the parity check bits, but more complicated). The received bits then can also be partitioned into the noisy $\boldsymbol{y}^a, \boldsymbol{y}^b$. Importantly, the code is designed so that the message can be decoded (albeit with errors) using either $\boldsymbol{y}^a$ or $\boldsymbol{y}^b$. The turbocoding algorithm then works as follows: It uses the model of $\boldsymbol{X}^a$ (trivial in this case) and of the channel noise to compute a posterior probability over $\boldsymbol{U}$ given $\boldsymbol{y}^a$. It then uses that posterior $\pi_a(U_1), \ldots, \pi_a(U_k)$ as a prior over $\boldsymbol{U}$ and computes a new posterior over $\boldsymbol{U}$, using the model for $\boldsymbol{X}^b$ and the channel, and $\boldsymbol{y}^b$ as the evidence, to compute a new posterior $\pi_b(U_1), \ldots, \pi_b(U_k)$. The "new information," which is $\pi_b(U_i)/\pi_a(U_i)$, is then transmitted back to the first decoder, and the process repeats until a stopping criterion is reached. In effect, the turbocoding idea was to use two weak coding schemes, but to "turbocharge" them using a feedback loop. Each decoder is used to decode one subset of received bits, generating a more informed distribution over the message bits to be subsequently updated by the other. The specific method proposed used particular coding scheme for the $\boldsymbol{X}^b$ bits, illustrated in figure 3b.

This process looked a lot like black magic, and in the beginning, many people did not even believe that the algorithm worked. However, when the empirical success of these properties was demonstrated conclusively, an attempt was made to understand its theoretical properties. McEliece, MacKay, and Cheng [1998] and Frey and MacKay [1997] subsequently showed that the specific message passing procedure proposed by Berrou et al. is precisely an application of belief propagation (with a particular message passing schedule) to the Bayesian network representing the turbocode (as in figure 3b).

4.3 Convergence of Loopy Belief Propagation

Pearl's main reason for rejecting the loopy belief propagation algorithm was the fact that it may fail to converge. Indeed, this is one of the thorniest issues associated with the use of belief propagation in practical applications — much more so than the fact that the resulting beliefs may not be exact. Nonconvergence is particularly problematic when we build systems that use inference as a subroutine within other tasks, for example, as the inner loop of a learning algorithm. Much work has been done on analyzing the convergence properties of generalized belief propagation algorithms, producing some valuable theoretical insights into its properties (such as the recent work of Ihler, Fisher, and Willsky [2005] and Mooij and Kappen [2007]). In practice, several approaches have been used for addressing the nonconvergence issue, some of which we now describe.

A first observation is that nonconvergence is often a local problem. In many practical cases, most of the beliefs in the network do converge, and only a small portion of the network remains problematic. In such cases, it is often quite reasonable simply to stop the algorithm at some point (for example, when some predetermined amount of time has elapsed) and use the beliefs at that point, or a running average of the beliefs over some time window. This heuristic is particularly reasonable when

we are not interested in individual beliefs, but rather in some aggregate over the entire network, for example, in a learning setting.

A second observation is that nonconvergence is often due to oscillations in the beliefs. As proposed by Murphy, Weiss, and Jordan [1999] and Heskes [2002], we can dampen the oscillations by reducing the difference between two subsequent updates. In particular, we can replace the update rule in (2) by a *smoothed* version that averages the update $\delta_{i\to j}$ with the previous message between the two cliques:

$$\delta_{i\to j} \leftarrow \lambda \left(\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \prod_{k \neq j} \delta_{k\to i} \right) + (1-\lambda)\delta_{i\to j}^{\text{old}}, \tag{4}$$

where λ is the damping weight and $\delta_{i\to j}^{\text{old}}$ is the previous value of the message. When $\lambda = 1$, this update is equivalent to standard belief propagation. For $0 < \lambda < 1$, the update is partial and although it shifts π_j toward agreement with π_i, it leaves some momentum for the old value of the belief, a dampening effect that in turn reduces the fluctuations in the beliefs. It turns out that this smoothed update rule is "equivalent" to the original update rule, in that a set of beliefs is a convergence point of the smoothed update if and only if it is a convergence point of standard updates. Moreover, one can show that, if run from a point close enough to a *stable convergence point* of the algorithm, with a sufficiently small λ, this smoothed update rule is guaranteed to converge. Of course, this guarantee is not very useful in practice, but there are indeed many cases where the smoothed update rule is convergent, whereas the original update rule oscillates indefinitely (see figure 4).

A broader-spectrum heuristic, which plays an important role not only in ensuring convergence but also in speeding it up considerably, is intelligent *message scheduling*. The simplest and perhaps most natural approach is to implement BP message passing as a *synchronous* algorithm, where all messages are updated at once. *Asynchronous* message passing updates messages one at a time, using the most recent version of the incoming messages to generate the outgoing message. It turns out that, in most cases, the synchronous schedule is far from optimal, both in terms of reaching convergence, and in the number of messages required for convergence. As one simple example, consider a cluster graph with m edges, and diameter d, synchronous message passing requires $m(d-1)$ messages to pass information from one side of the graph to the other. By contrast, *asynchronous* message passing, appropriately scheduled, can pass information between two clusters at opposite ends of the graph using $d-1$ messages. Moreover, the fact that, in synchronous message passing, each cluster uses messages from its neighbors that are based on their previous beliefs appears to increase the chances of oscillatory behavior and nonconvergence in general.

In practice, an *asynchronous* message passing schedule works significantly better than the synchronous approach (see figure 4). Moreover, even greater improvements can be obtained by scheduling messages in a guided way. One approach, called *tree*

reparameterization (TRP) [Wainwright, Jaakkola, and Willsky 2003], selects a set of trees, each of which spans a large number of the clusters, and whose union covers all of the edges in the network. The TRP algorithm then iteratively selects a tree and does an upward-downward calibration of the tree, keeping all other messages fixed. Of course, calibrating this tree has the effect of "uncalibrating" other trees, and so this process repeats. This approach has the advantage of passing information more globally within the graph. It therefore converges more often, and more quickly, than other asynchronous schedules, particularly if the trees are selected using a careful design that accounts for the properties of the problem.

An even more flexible approach attempts to detect dynamically in which parts of the network messages would be most useful. Specifically, as we observed, often some parts of the network converge fairly quickly, whereas others require more messages. We can schedule messages in a way that accounts for their potential usefulness; for example, we can pass a message between clusters where the beliefs disagree most strongly on the sepset. This approach, called *residual belief propagation* [Elidan, McGraw, and Koller 2006] is convenient, since it is fully general and does not require a deep understanding of the properties of the network. It also works well across a range of different real-world networks.

To illustrate these issues, we show the behavior of loopy belief propagation on an 11×11 grid with binary-valued variables; the network is parameterized as an *Ising model* — one where the pairwise potentials are defined as: $\phi_{i,j}(x_i, x_j) = \exp^{w_{i,j} x_i x_j}$. The network potentials were randomly sampled as follows: Each univariate potential was sampled uniformly in the interval $[0, 1]$; for each pair of variables X_i, Z_j, $w_{i,j}$ is sampled uniformly in the range $[-C, C]$. This sampling process creates an energy function where some potentials are attractive ($w_{i,j} > 0$), causing adjacent variables to prefer taking the same value, and some are repulsive ($w_{i,j} < 0$). This regime can result in very difficult inference problems. The magnitude of C (11 in this example) controls the magnitude of the forces and higher values correspond, on average, to more challenging inference problems.

Figure 4 illustrates the convergence behavior on this problem. (a) shows the percentage of messages converged as a function of time for three variants of the belief propagation algorithm: synchronous BP with smoothing (dashed line), where only a small fraction of the messages ever converge; asynchronous BP with smoothing that converges (solid line); asynchronous BP with no smoothing (dash-dot line) that does not fully converge. The benefit of using asynchronous propagation over synchronous updating is obvious. At early round, smoothing tends to slow convergence, because some messages converge quickly when updates are not slowed down by smoothing. However, as the algorithm progresses, smoothing allows all messages to achieve convergence, whereas the unsmoothed algorithm never converges. We note that smoothing is equally beneficial for synchronous updates; indeed, the graph for unsmoothed synchronous updates is not shown because virtually none of the messages achieve convergence.

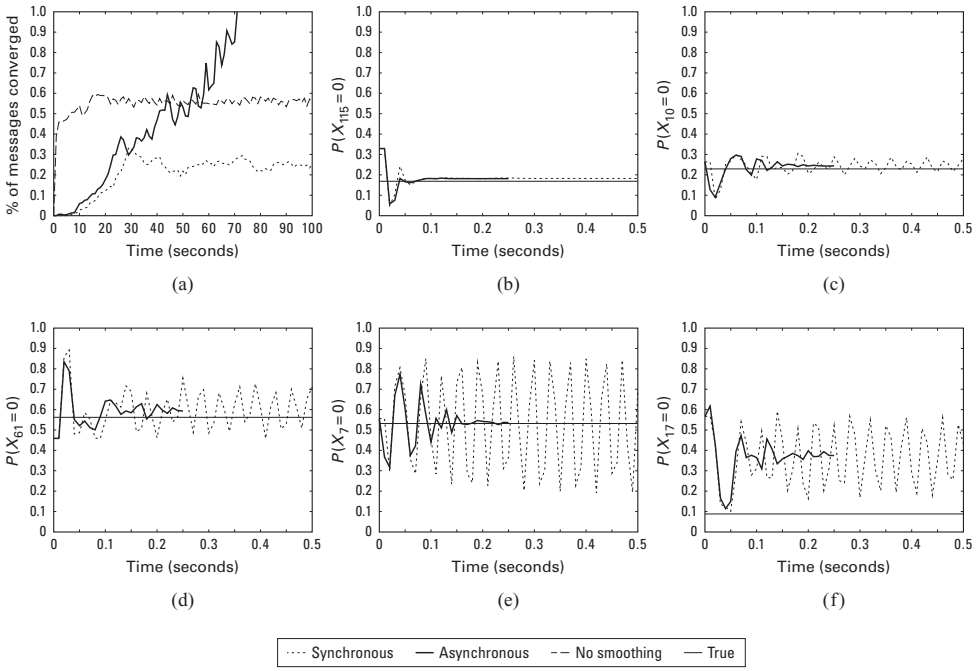


Figure 4. **Example of behavior of BP in practice on an** 11×11 **Ising grid.** Comparison of three different BP variants: synchronous BP with smoothing (dashed line), asynchronous BP with smoothing (solid line), and asynchronous BP with no smoothing (dash-dot line — only shown in (a)). (a) Percentage of messages converged as a function of time. (b) A marginal where both variants converge rapidly. (c–e) Marginals where the synchronous BP marginals oscillate around the asynchronous BP marginals. (f) A marginal where both variants are inaccurate.

The remaining panels illustrate the progression of the marginal beliefs over the course of the algorithm. (b) shows a marginal where both the synchronous and asynchronous updates converge quite rapidly and are close to the true marginal (thin solid black). Such behavior is atypical, and it comprises only around 10 percent of the marginals in this example. In the vast majority of the cases (almost 80 percent in this example), the synchronous beliefs oscillate around the asynchronous ones ((c)–(e)). In many cases, such as the ones shown in (e), the entropy of the synchronous beliefs is quite significant. For about 10 percent of the marginals (for example (f)), both the asynchronous and synchronous marginals are inaccurate. In these cases, using more informed message schedules can significantly improve the algorithms performance.

These qualitative differences between the BP variants are quite consistent across many random and real-life models. Typically, the more complex the inference problem, the larger the gaps in performance. For very complex real-life networks in-

volving tens of thousands of variables and multiple cycles, even asynchronous BP is not very useful and more elaborate propagation methods or convergent alternatives must be adopted.

5 Max-Product Message Passing for MAP Inference

We now consider the application of belief propagation algorithms to the task of computing the MAP assignment, as in (1).

5.1 Computing Max-Marginals

The MAP task goes hand in hand with finding the value of the unnormalized probability of the most likely assignment: $\max_{\boldsymbol{x}} \tilde{P}_\Phi(\boldsymbol{x})$. We note that, given an assignment $\boldsymbol{x}$, we can easily compute its unnormalized probability simply by multiplying all of the factors in Φ, evaluated at $\boldsymbol{x}$. However, we cannot retrieve the *actual* probability of $\boldsymbol{x}$ without computing the partition function, a problem that requires that we also solve the sum-product task. Because $\tilde{P}_\Phi$ is a product of factors, tasks that involve maximizing $\tilde{P}_\Phi$ are often called *max-product* inference tasks.

A large subset of algorithms for the MAP problem operate by first computing a set of factors that are *max-marginals*. For a general function f, we define the *max-marginal* of f relative to a set of variables $\boldsymbol{Y}$ as

$$MaxMarg_f(\boldsymbol{y}) = \max_{\boldsymbol{x}\langle \boldsymbol{Y}\rangle = \boldsymbol{y}} f(\boldsymbol{x}), \tag{5}$$

for any assignment $\boldsymbol{y} \in Val(\boldsymbol{Y})$. For example, the max-marginal $MaxMarg_{\tilde{P}_\Phi}(\boldsymbol{Y})$ is a factor that determines a value for each assignment $\boldsymbol{y}$ to $\boldsymbol{Y}$; this value is the unnormalized probability of the most likely joint assignment consistent with $\boldsymbol{y}$.

The same belief propagation algorithms that we showed for sum-product can easily be adapted to the case of max-product. In particular, the *max-product belief propagation* algorithm in clique trees executes precisely the same initialization and overall message scheduling as in the sum-product clique tree algoirthm; the only difference is that we replace (2) with the following:

$$\delta_{i\to j} = \max_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \pi_i^0 \times \prod_{k \in (\mathcal{N}_i - \{j\})} \delta_{k\to i}. \tag{6}$$

As for sum-product message passing, the algorithm will converge after a single upward and downward pass. After those steps, the resulting clique tree $\mathcal{T}$ will contain the appropriate max-marginal in every clique. In particular, for each clique $\boldsymbol{C}_i$ and each assignment $\boldsymbol{c}_i$ to $\boldsymbol{C}_i$, we will have that

$$\pi_i[\boldsymbol{c}_i] = MaxMarg_{\tilde{P}_\Phi}(\boldsymbol{c}_i). \tag{7}$$

That is, the clique belief contains, for each assignment $\boldsymbol{c}_i$ to the clique variables, the (unnormalized) measure $\tilde{P}_\Phi(\boldsymbol{x})$ of the most likely assignment $\boldsymbol{x}$ consistent with $\boldsymbol{c}_i$. Note that, because the max-product message passing process does not compute

the partition function, we *cannot* derive from these max-marginals the actual probability of any assignment; however, because the partition function is a constant, we can still compare the values associated with different assignments, and therefore compute the assignment $\boldsymbol{x}$ that maximizes $\tilde{P}_\Phi(\boldsymbol{x})$.

Because max-product message passing over a clique tree produces max-marginals in every clique, and because max-marginals must agree, it follows that any two adjacent cliques must agree on their sepset:

$$\max_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \pi_i = \max_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \pi_j = \mu_{i,j}(\boldsymbol{S}_{i,j}). \tag{8}$$

In this case, the clusters are said to be *max-calibrated*. We say that a clique tree is *max-calibrated* if all pairs of adjacent cliques are max-calibrated.

The same transformation from sum-product to max-product can be applied to the case of loopy belief propagation. Here, the algorithm is the same as in figure 2, except that we replace the sum-product message computation with the max-product computation of (6). As for sum-product, there are no guarantees that this algorithm will converge. Indeed, in practice, it tends to converge somewhat less often than the sum-product algorithm, perhaps because the averaging effect of the summation operation tends to smooth out messages, and reduce oscillations. The same ideas that we discussed in section 4.3 can be used to improve convergence in this algorithm as well.

At convergence, the result will be a set of calibrated clusters: As for sum-product, if the clusters are not calibrated, convergence has not been achieved, and the algorithm will continue iterating. However, the resulting beliefs will not generally be the exact max-marginals; these beliefs are often called *pseudo-max-marginals*.

5.2 Locally Optimal Assignments

How do we go from a set of (approximate) max-marginals to a consistent joint assignment that has high probability? One obvious solution is to use the max-marginal for each variable X_i to compute its own optimal assignment, and thereby compose a full joint assignment to all variables. However, this simplistic approach may not always work, even if we have exact max-marginals.

EXAMPLE 3. Consider a simple XOR-like distribution $P(X_1, X_2)$ that gives probability 0.1 to the assignments where $X_1 = X_2$ and 0.4 to the assignments where $X_1 \neq X_2$. In this case, for each assignment to X_1, there is a corresponding assignment to X_2 whose probability is 0.4. Thus, the max-marginal of X_1 is the symmetric factor $(0.4, 0.4)$, and similarly for X_2. Indeed, we can choose either of the two values for X_1 and complete it to a MAP assignment, and similarly for X_2. However, if we choose the values for X_1 and X_2 in an *inconsistent* way, we may get an assignment whose probability is much lower. Thus, our joint assignment cannot be chosen by separately optimizing the individual max-marginals.

Such examples cannot arise if the max-marginals are *unambiguous*: For each

variable X_i, there is a unique x_i^* that maximizes:

$$x_i^* = \max_{x_i \in Val(X_i)} MaxMarg_f(x_i). \tag{9}$$

This condition prevents symmetric cases like the one in the preceding example. Indeed, it is not difficult to show that the following two conditions are equivalent:

- The set of node beliefs $\{MaxMarg_{\tilde{P}_\Phi}(X_i) \ : \ X_i \in \mathcal{X}\}$ is unambiguous, with

  $$x_i^* = \text{argmax}_{x_i} MaxMarg_{\tilde{P}_\Phi}(X_i)$$

  the unique optimizing value for X_i;
- $\tilde{P}_\Phi$ has a unique MAP assignment $(x_1^*, \ldots, x_n^*)$.

For generic probability measures, the assumption of unambiguity is not overly stringent, since we can always break ties by introducing a slight random perturbation into all of the factors, making all of the elements in the joint distribution have slightly different probabilities. However, if the distribution has special structure — deterministic relationships or shared parameters — that we want to preserve, this type of ambiguity may be unavoidable.

The situation where there are ties in the node beliefs is more complex. In this case, we say that an assignment $\boldsymbol{x}^*$ has the *local optimality property* if, for each cluster $\boldsymbol{C}_i$ in the tree, we have that

$$\boldsymbol{x}^* \langle \boldsymbol{C}_i \rangle \in \text{argmax}_{\boldsymbol{c}_i} \pi_i[\boldsymbol{c}_i], \tag{10}$$

that is, the assignment to $\boldsymbol{C}_i$ in $\boldsymbol{x}^*$ optimizes the $\boldsymbol{C}_i$ belief. The task of finding a locally optimal assignment $\boldsymbol{x}^*$ given a max-calibrated set of beliefs is called the *decoding* task.

Importantly, for approximate max-marginals derived from loopy belief propagation, a locally optimal joint assignment may not exist:

EXAMPLE 4. Consider a cluster graph with the three clusters $\{A, B\}, \{B, C\}, \{A, C\}$ and the beliefs

	a^1	a^0
b^1	1	2
b^0	2	1

	b^1	b^0
c^1	1	2
c^0	2	1

	a^1	a^0
c^1	1	2
c^0	2	1

These beliefs are max-calibrated, in that all messages are $(2, 2)$. However, there is no joint assignment that maximizes all of the cluster beliefs simultaneously. For example, if we select a^0, b^1, we maximize the value in the A, B belief. We can now select c^0 to maximize the value in the B, C belief. However, we now have a nonmaximizing assignment a^0, c^0 in the A, C belief. No matter which assignment of values we select in this example, we do not obtain a single joint assignment that maximizes all three beliefs. Loops such as this are often called *frustrated*.

How do we find a locally optimal joint assignment, if one exists? Recall from the definition that an assignment is locally optimal if and only if it selects one of the optimizing assignments in every single cluster. Thus, we can essentially label the assignments in each cluster as either "legal" if they optimize the belief or "illegal" if they do not. We now must search for an assignment to $\mathcal{X}$ that results in a legal value for each cluster. This problem is precisely an instance of a *constraint satisfaction problem (CSP)*. A constraint satisfaction problem can be defined in terms of a Markov network (or factor graph) where all of the entries in the beliefs are either 0 or 1. The CSP problem is now one of finding an assignment whose (unnormalized) measure is 1, if one exists, and otherwise reporting failure. In other words, the CSP problem is simply that of finding the MAP assignment in this model with $\{0, 1\}$-valued beliefs. The field of CSP algorithms is a large one, and a detailed survey is outside the scope of the paper; see Dechter [2003] for a recent survey. Interestingly, it is an area to which Pearl also made important early contributions [Dechter and Pearl 1987]. Recent work has reinvigorated this trajectory, studying the surprisingly deep connections between CSP methods and belief propagation, and exploiting it (for example, within the context of the *survey propagation* algorithm [Maneva, Mossel, and Wainwright 2007]).

Thus, given a max-product calibrated cluster graph, we can convert it to a discrete-valued CSP by simply taking the belief in each cluster, changing each assignment that locally optimizes the belief to 1 and all other assignments to 0. We then run some CSP solution method. If the outcome is an assignment that achieves 1 in every belief, this assignment is guaranteed to be a locally optimal assignment. Otherwise, there is no locally optimal assignment. Importantly, as we discuss below, for the case of calibrated clique trees, we are guaranteed that this approach finds a globally optimal assignment.

In the case where there is no locally optimal assignment, we must resort to the use of alternative solution methods. One heuristic in this latter situation is to use information obtained from the max-product propagation to construct a partial assignment. For example, assume that a variable X_i is unambiguous in the calibrated cluster graph, so that the only value that locally optimizes its node marginal is x_i. In this case, we may decide to restrict attention only to assignments where $X_i = x_i$. In many real-world problems, a large fraction of the variables in the network are unambiguous in the calibrated max-product cluster graph. Thus, this heuristic can greatly simplify the model, potentially even allowing exact methods (such as clique tree inference) to be used for the resulting restricted model. We note, however, that the resulting assignment would not necessarily satisfy the local optimality condition, and all of the guarantees we will present hold only under that assumption.

5.3 Optimality Guarantees

The local optimality property comes with some fairly strong guarantees. In particular, for exact max-marginals, one can show the following result:

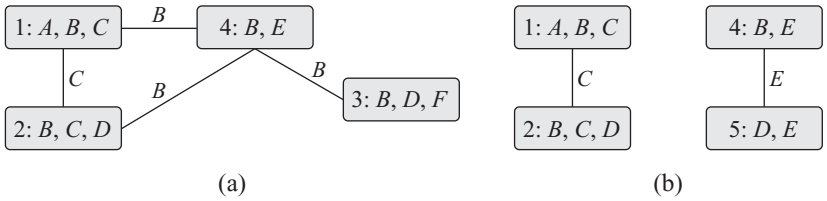


Figure 5. **Two induced subgraphs derived from figure 1a.** (a) Graph over $\{B, C\}$; (b) Graph over $\{C, E\}$.

THEOREM 5. *Let $\pi_i[\boldsymbol{C}_i]$ be a set of max-marginals for the distribution $\tilde{P}_\Phi$, and let $\mu_{i,j}$ be the associated sepset beliefs. Then an assignment $\boldsymbol{x}^*$ satisfies the local optimality property relative to the beliefs $\{\pi_i[\boldsymbol{C}_i]\}_{i \in \mathcal{V}_\mathcal{T}}$ if and only if it is the global MAP assignment relative to $\tilde{P}_\Phi$.*

What type of guarantee can we provide for a decoded assignment from the pseudo-max-marginals produced by the max-product belief propagation algorithm? It is certainly not the case that this assignment is the MAP assignment; nor is it even the case that we can guarantee that the probability of this assignment is "close" in any sense to that of the true MAP assignment. However, if we can construct a locally optimal assignment $\boldsymbol{x}^*$ relative to the beliefs produced by max-product BP, we can prove that $\boldsymbol{x}^*$ is a *strong local maximum*, in the following sense: For certain subsets of variables $\boldsymbol{Y} \subset \mathcal{X}$, there is no assignment $\boldsymbol{x}'$ that is higher-scoring than $\boldsymbol{x}^*$ and differs from it only in the assignment to $\boldsymbol{Y}$. These subsets $\boldsymbol{Y}$ are those that induce any disjoint union of subgraphs each of which contains at most a single loop (including trees, which contain no loops).

More precisely, for a subset of variables $\boldsymbol{Y}$, we define the *induced subgraph* $\mathcal{U}_{\boldsymbol{Y}}$ to be the subgraph of clusters and sepsets in $\mathcal{U}$ that contain some variable in $\boldsymbol{Y}$. In the straightforward cluster graph for a pairwise Markov network (as described earlier), the induced subgraph for a set $\boldsymbol{Y}$ is simply the set of nodes corresponding to $\boldsymbol{Y}$ and any edges that contain them. Figure 5 shows two examples of an induced subgraph for a more general cluster graph.

We can now state the following important theorem:

THEOREM 6. *Let $\mathcal{U}$ be a max-product calibrated cluster graph for $\tilde{P}_\Phi$, and let $\boldsymbol{x}^*$ be a locally optimal assignment for $\mathcal{U}$. Let $\boldsymbol{Z}$ be any set of variables for which $\mathcal{U}_{\boldsymbol{Z}}$ is a collection of disjoint subgraphs each of which contains at most a single loop. Then for any assignment $\boldsymbol{x}'$ which is the same as $\boldsymbol{x}^*$ except for the assignment to the variables in $\boldsymbol{Z}$, we have that $\tilde{P}_\Phi(\boldsymbol{x}') \leq \tilde{P}_\Phi(\boldsymbol{x}^*)$.*

This result generalizes one by Weiss and Freeman [2001], who showed a corresponding version in the unambiguous case, for a pairwise Markov network. Its proof in the more general case rests heavily on the analysis of Wainwright, Jaakkola, and Willsky [2005], who proved that a different variant of max-product message passing,

if it converges to an unambiguous solution, is guaranteed to produce the true MAP assignment.

This theorem implies as a corollary the (well-known) result that, for a max-product calibrated clique tree, the decoding process is guaranteed to produce a globally optimal (MAP) assignment. However, its more important implications are in the context of a loopy cluster graph.

EXAMPLE 7. Consider a 4×4 grid network, and assume that we use the pairwise cluster graph construction described earlier. In this case, theorem 6 implies that the MAP solution found by max-product belief propagation has higher probability than any assignment obtained by changing the assignment to any of the following subsets of variables $\boldsymbol{Y}$: a set of variables in any single row, such as $\boldsymbol{Y} = \{A_{1,1}, A_{1,2}, A_{1,3}, A_{1,4}\}$; a set of variables in any single column; a "comb" structure such as the variables in row 1, column 2 and column 4; a single loop, such as $\boldsymbol{Y} = \{A_{1,1}, A_{1,2}, A_{2,2}, A_{2,1}\}$; or a collection of disconnected subsets of the preceding form.

This result is a powerful one, inasmuch as it shows that the solution obtained from max-product belief propagation is robust against large perturbations. Thus, although one can construct examples where max-product belief propagation obtains the wrong solutions, these solutions are strong local maxima, and therefore they often have high probability. Conversely, it is important to realize the limitations of this result. For one, it only applies if the max-product belief propagation algorithm converges to a fixed point, which is not always the case; indeed, as we mentioned earlier, convergence here is generally harder to achieve than in the sum-product variant. Second, even if convergence is achieved, one has to be able to decode the resulting pseudo-max-marginals in order to obtain a locally-optimal joint assignment. It is only if these two conditions hold that this result can be brought to bear.

6 Conclusions

This paper has reviewed a small fraction of the recent results regarding the belief propagation algorithm. This line of work has been hugely influential in the area of probabilistic modeling, both in practice and in theory. On the practical side, belief propagation algorithms are among the most commonly used for inference in graphical models for which exact inference is intractable. They have been used successfully for a broad range of applications, including message decoding, natural language processing, computer vision, computational biology, web analysis, and many more. There have also been tremendous developments on the algorithmic side, with many important extensions to the basic approach.

On the theoretical side, the work of many people has served to provide a much deeper understanding of the theoretical foundations of this algorithm, which has tremendously influenced our entire perspective on probabilistic inference. One sem-

inal line of work along these lines was initiated by the landmark paper of Yedidia, Freeman, and Weiss [2000, 2005], showing that beliefs obtained as fixed points of the belief propagation algorithm are also solutions to an optimization problem; this problem is an approximation to another optimization problem whose solutions are the exact marginals that would be obtained from clique tree inference. Thus, both exact (clique tree) and approximate (cluster graph) inference can be viewed in terms of optimization of an objective. This observation was the basis for the development of a whole range of novel methods that explored different variations on the formulation of the optimization problem, or different algorithms for performing the optimization. One such line of work uses *convex* versions of the optimization problem underlying belief propagation, a trajectory initiated by Wainwright, Jaakkola, and Willsky [2002]. Algorithms based on this approach (e.g., [Heskes 2006; Hazan and Shashua 2008]) can also guarantee convergence as well as provide bounds on the partition function.

For the MAP problem, a similar optimization-based view has also recently come to dominate the field. Here, the original MAP problem is reformulated as an *integer programming* problem, where the (discrete-valued) variables in the optimization represent the space of possible assignments $\boldsymbol{x}$. This discrete optimization is then relaxed to produce a continuous-valued optimization problem that is a *linear program* (LP). This LP-relaxation approach was first proposed by Schlesinger [1976], and then subsequently rediscovered independently by several researchers. Most notably, Wainwright, Jaakkola, and Willsky [2005] established the first connection between the dual problem to this LP and message passing algorithms, and proposed a new message-passing algorithm (TRW) based on this connection. Many recent works build on these ideas and develop a suite of increasingly better algorithms for solving the MAP inference problem. Some of these algorithms utilize message-passing techniques; others merely adopt the idea of using the LP dual but utilize other optimization methods for solving it. Importantly, for several of these algorithms, one can guarantee that a solution, if one is found, is guaranteed to be the optimal MAP assignment.

In summary, the simple message-passing algorithm first proposed by Pearl has recently returned to revolutionize the world of inference in graphical models. It has dramatically affected both the practice in the field and has led to a new, optimization-based perspective on the foundations of the inference task. This new understanding has, in turn, given rise to the development of much better algorithms, which continue to improve our ability to apply probabilistic graphical models to challenging, real-world applications.

Acknowledgments This material in this review paper is extracted from the book of Koller and Friedman [2009], published by MIT Press. Some of this material is based on contributions by Nir Friedman and Gal Elidan. I also thank Amir Globerson, David Sontag, and Yair Weiss for useful discussions regarding MAP inference.

References

Berrou, C., A. Glavieux, and P. Thitimajshima (1993). Near Shannon limit error-correcting coding: Turbo codes. In *Proc. International Conference on Communications*, pp. 1064–1070.

Cooper, G. (1990). Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence 42*, 393–405.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Dechter, R., K. Kask, and R. Mateescu (2002). Iterative join-graph propagation. In *Proc. 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 128–136.

Dechter, R. and J. Pearl (1987). Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence 34*(1), 1–38.

Elidan, G., I. McGraw, and D. Koller (2006). Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proc. 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Frey, B. and D. MacKay (1997). A revolution: Belief propagation in graphs with cycles. In *Proc. 11th Conference on Neural Information Processing Systems (NIPS)*.

Hazan, T. and A. Shashua (2008). Convergent message-passing algorithms for inference over general graphs with convex free energies. In *Proc. 24th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Heckerman, D., E. Horvitz, and B. Nathwani (1992). Toward normative expert systems: Part I. The Pathfinder project. *Methods of Information in Medicine 31*, 90–105.

Heskes, T. (2002). Stable fixed points of loopy belief propagation are minima of the Bethe free energy. In *Proc. 16th Conference on Neural Information Processing Systems (NIPS)*, pp. 359–366.

Heskes, T. (2006). Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Machine Learning Research 26*, 153–190.

Ihler, A. T., J. W. Fisher, and A. S. Willsky (2005). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research 6*, 905–936.

Jensen, F. V., K. G. Olesen, and S. K. Andersen (1990, August). An algebra of Bayesian belief universes for knowledge-based systems. *Networks 20*(5), 637–659.

Kim, J. and J. Pearl (1983). A computational model for combined causal and diagnostic reasoning in inference systems. In *Proc. 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 190–193.

Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

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

Maneva, E., E. Mossel, and M. Wainwright (2007, July). A new look at survey propagation and its generalizations. *Journal of the ACM 54*(4), 2–41.

McEliece, R., D. MacKay, and J.-F. Cheng (1998, February). Turbo decoding as an instance of Pearl's "belief propagation" algorithm. *IEEE Journal on Selected Areas in Communications 16*(2).

Mooij, J. M. and H. J. Kappen (2007). Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. Information Theory 53*, 4422–4437.

Murphy, K. P., Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: an empirical study. In *Proc. 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 467–475.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. San Mateo, California: Morgan Kaufmann.

Schlesinger, M. (1976). Sintaksicheskiy analiz dvumernykh zritelnikh singnalov v usloviyakh pomekh (syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika 4*, 113–130.

Shafer, G. and P. Shenoy (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence 2*, 327–352.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal 27*, 379–423; 623–656.

Shimony, S. (1994). Finding MAPs for belief networks in NP-hard. *Artificial Intelligence 68*(2), 399–410.

Shwe, M., B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods of Information in Medicine 30*, 241–55.

Szeliski, R., R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother (2008, June). A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. on Pattern Analysis and Machine Intelligence 30*(6), 1068–1080. See `http://vision.middlebury.edu/MRF` for more detailed results.

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

Wainwright, M., T. Jaakkola, and A. Willsky (2005). MAP estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*.

Wainwright, M., T. Jaakkola, and A. S. Willsky (2002). A new class of upper bounds on the log partition function. In *Proc. 18th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Weiss, Y. (1996). Interpreting images by propagating bayesian beliefs. In *Proc. 10th Conference on Neural Information Processing Systems (NIPS)*, pp. 908–914.

Weiss, Y. and W. Freeman (2001). On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory 47*(2), 723–735.

Yedidia, J., W. Freeman, and Y. Weiss (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. Information Theory 51*, 2282–2312.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2000). Generalized belief propagation. In *Proc. 14th Conference on Neural Information Processing Systems (NIPS)*, pp. 689–695.

13

Extending Bayesian Networks to the Open-Universe Case

Brian Milch and Stuart Russell

1 Introduction

One of Judea Pearl's now-classic examples of a Bayesian network involves a home alarm system that may be set off by a burglary or an earthquake, and two neighbors who may call the homeowner if they hear the alarm. Like most scenarios modeled with BNs, this example involves a known set of objects (one house, one alarm, and two neighbors) with known relations between them (the alarm is triggered by events that affect this house; the neighbors can hear this alarm). These objects and relations determine the relevant random variables and their dependencies, which are then represented by nodes and edges in the BN.

In many real-world scenarios, however, the relevant objects and relations are initially unknown. For instance, suppose we have a set of ASCII strings containing irregularly formatted and possibly erroneous academic citations extracted from online documents, and we wish to make a list of the distinct publications that are referred to, with correct author names and titles. In this case, the publications, authors, venues, and so on are not known in advance, nor is the mapping between publications and citations. The same challenge of making inferences about unknown objects is called *coreference resolution* in natural language processing, *data association* in multitarget tracking, and *record linkage* in database systems. The issue is actually much more widespread than this short list suggests; it arises in any data interpretation problem in which objects or events come without unique identifiers.

In this chapter, we show how the Bayesian network (BN) formalism that Judea Pearl pioneered has been extended to handle such scenarios. The key contribution on which we build is the use of acyclic directed graphs of local conditional distributions to generate well-defined, global probability distributions. We begin with a review of *relational probability models* (RPMs), which specify how to construct a BN for a given set of objects and relations. We then describe *open-universe probability models*, or OUPMs, which represent uncertainty about what objects exist. OUPMs may not boil down to finite, acyclic BNs; we present results from Milch [2006] showing how to extend the factorization and conditional independence semantics of BNs to models that are only *context-specifically* finite and acyclic. Finally, we discuss how Markov chain Monte Carlo (MCMC) methods can be used to perform approximate inference on OUPMs and briefly describe some applications.

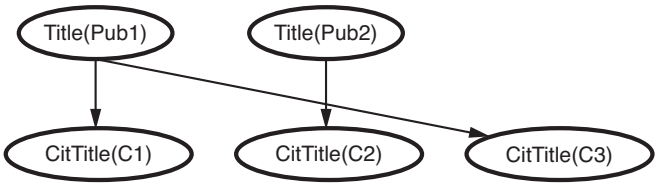


Figure 1. A BN for a bibliography scenario where we know that citations Cit1 and Cit3 refer to Pub1, while citation Cit2 refers to Pub2.

$$\mathsf{Title}(p) \sim \mathsf{TitlePrior}()$$

$$\mathsf{CitTitle}(c) \sim \mathsf{TitleEditCPD}(\mathsf{Title}(\mathsf{PubCited}(c)))$$

Figure 2. Dependency statements for a bibliography scenario, where p ranges over publications and c ranges over citations. This model assumes that the PubCited function and the sets of publications and citations are known.

2 Relational probability models

Suppose we are interested in inferring the true titles of publications given some observed citations, and we know the set of publications and the mapping from citations to publications. Assuming we have a prior distribution for true title strings (perhaps a word n-gram model) and a conditional probability distribution (CPD) for citation titles given true titles, we can construct a BN for this scenario, as shown in Figure 1.

2.1 The RPM formalism

A relational probability model represents such a BN compactly using *dependency statements* (see Figure 2), which specify the CPDs and parent sets for whole classes of variables at once. In this chapter, we will not specify any particular syntax for dependency statements, although we use a syntax based loosely on BLOG [Milch et al. 2005]. The important point is that dependencies are specified via relations among objects. For example, the dependency statement for CitTitle in Figure 2 specifies that each $\mathsf{CitTitle}(c)$ variable depends (according to the conditional distribution TitleEditCPD that describes how titles may be erroneously transcribed) on $\mathsf{Title}(\mathsf{PubCited}(c))$—that is, on the true title of the publication that c cites. The PubCited relation is nonrandom, and thus forms part of the known *relational skeleton* of the RPM. In this case, the skeleton also includes the sets of citations and publications.

Formally, it is convenient to think of an RPM M as defining a probability distribution over a set of model structures of a typed first-order logical language. These structures are called the *possible worlds* of M and denoted Ω_M. The function sym-

bols of the logical language (including constant and predicate symbols) are divided into a set of *nonrandom* function symbols whose interpretations are specified by the relational skeleton, and a set of *random* function symbols whose interpretations vary between possible worlds. An RPM includes one dependency statement for each random function symbol.

Each RPM M defines a set of *basic random variables* $\mathbf{V}_M$, one for each application of a random function to a tuple of arguments. We will write $X(\omega)$ for the value of a random variable X in world ω. If X represents the value of the random function f on some arguments, then the dependency statement for f defines a parent set and CPD for X. The parent set for X, denoted $\mathrm{Pa}(X)$, is the set of basic variables that are needed to evaluate the expressions in the dependency statement in any possible world. For instance, if we know that $\mathsf{PubCited}(\mathsf{Cit1}) = \mathsf{Pub1}$, then the dependency statement in Figure 2 yields the single parent $\mathsf{Title}(\mathsf{Pub1})$ for the variable $\mathsf{CitTitle}(\mathsf{Cit1})$. The CPD for a basic variable X is a function $\varphi_X(x, \mathbf{pa})$ that defines a conditional probability distribution over values x of X given each instantiation $\mathbf{pa}$ of $\mathrm{Pa}(X)$. We obtain this CPD by evaluating the expressions in the dependency statement (such as $\mathsf{Title}(\mathsf{PubCited}(\mathsf{Cit1}))$) and passing them to an *elementary distribution function* such as $\mathsf{TitleEditCPD}$.

Thus, an RPM defines a BN over its basic random variables. If this BN is acyclic, it defines a joint distribution for the basic RVs. Since there is a one-to-one correspondence between full instantiations of $\mathbf{V}_M$ and worlds in Ω_M, this BN also gives us a probability measure over Ω_M. We define this to be the probability measure represented by the RPM.

2.2 Relational uncertainty

This formalism also allows us to model cases of *relational uncertainty*, such as a scenario where the mapping from citations to publications is unknown. We can handle this by making $\mathsf{PubCited}$ a random function and giving it a dependency statement such as:

$$\mathsf{PubCited}(c) \sim \mathsf{Uniform}(\{\mathsf{Pub}\ p\}) \ .$$

This statement says that each citation refers to a publication chosen uniformly at random from the set of all publications p. The dependency statement for $\mathsf{CitTitle}$ in Figure 2 now represents a *context-specific* dependency: for a given citation C_i, the $\mathsf{Title}(p)$ variable that $\mathsf{CitTitle}(C_i)$ depends on varies from world to world.

In the BN defined by this model, shown in Figure 3, the parents of each $\mathsf{CitTitle}(c)$ variable include all variables that might be needed to evaluate the dependency statement for $\mathsf{CitTitle}(c)$ in any possible world. This includes $\mathsf{PubCited}(c)$ and all the $\mathsf{Title}(p)$ variables. The CPD in the BN is a multiplexer that conditions on the appropriate $\mathsf{Title}(p)$ variable for each value of $\mathsf{PubCited}(c)$. If the BN constructed this way is still finite and acyclic, the usual BN semantics hold.

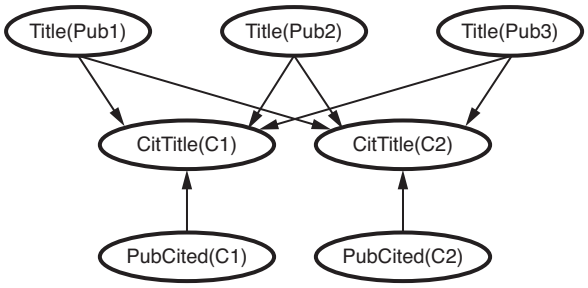


Figure 3. A BN arising from an RPM with relational uncertainty.

2.3 Names, objects, and identity uncertainty

We said earlier that the function symbols of an RPM include the constants and predicate symbols. For predicates, this simply means that a predicate can be thought of as a Boolean function that returns *true* or *false* for each tuple of arguments. The constants, on the other hand, are 0-ary functions that refer to objects. In most RPM languages, all constants are nonrandom and assumed to refer to distinct objects—the *unique names* assumption for constants. With this assumption, there is no need to distinguish between constant symbols and the objects they refer to, which is why we are able to name the basic random variables Title(Pub1), Title(Pub2) and so on, even though, strictly speaking, the arguments should be objects in the domain rather than constant symbols.

If the RPM language allows constants to be random functions, then the equivalent BN will include a node for each such constant. For example, suppose that Milch asks Russell to "fix the typo in the Pfeffer citation." Russell's mental software may already have formed nonrandom constant symbols C1, C2, and so on for all the citations at the end of the chapter, and these are in one-to-one correspondence with all the objects in this particular universe. It may then form a new constant symbol ThePfefferCitation, which co-refers with one of these. Because there is more than one citation to a work by Pfeffer, there is *identity uncertainty* concerning which citation object the new symbol refers to. Identity uncertainty is a degenerate form of relational uncertainty, but often has a quite distinct flavor.

3 Open-universe probability models

For all RPMs, even those with relational and identity uncertainty, the *objects* are known and are the same across all possible worlds. If the set of objects is unknown, however—e.g., if we don't know the set of publications that exist and might be cited—then RPMs as we have described them do not suffice. Whereas an RPM can be seen as defining a generative process that chooses a value for each random function on each tuple of arguments, an *open-universe probability model* (OUPM) includes generative steps that add objects to the world. These steps set the values of *number variables*.

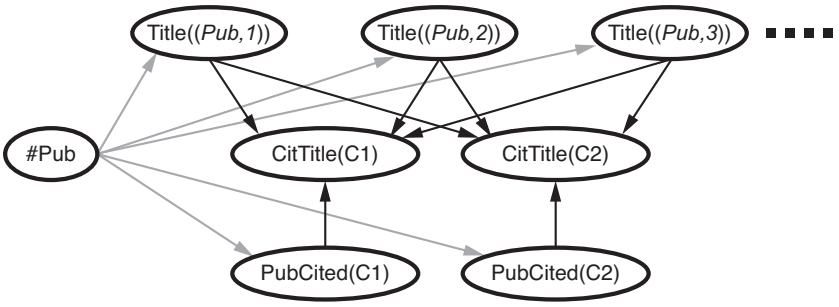


Figure 4. A BN that defines a probability distribution over worlds with unbounded numbers of publications.

3.1 Number variables

For the bibliography example, we introduce just one number variable, defining the total number of publications. There is no reason to place any *a priori* upper bound on the number of publications; we might be interested in asking how many publications there are for which we have found no citations (this question becomes more well-defined and pressing if we ask, say, how many aircraft are in an area but have not generated a blip on our radar screens). Thus, this number variable may have a distribution that assigns positive probability to all natural numbers.

We can specify the conditional probability distribution for a number variable using a dependency statement. In our bibliography example, we might use a very simple statement:

$$\#\mathsf{Pub} \sim \mathsf{NumPubsPrior}() \ .$$

Number variables can also depend on other variables; we will consider an example of this below.

In the RPM where we had a fixed set of publications, the relational skeleton specified a constant symbol such as Pub1 for each publication. In an OUPM where the set of publications is unknown, it does not make sense for the language to include such constant symbols. The possible worlds contain publication objects—which will assume are pairs $\langle Pub, 1\rangle$, $\langle Pub, 2\rangle$, etc.—but now they are not necessarily in one-to-one correspondence with any constant symbols.

The set of basic variables now includes the number variable #Pub itself, and variables for the application of each random function to all arguments that exist in any possible world. Figure 4 shows the BN over these variables. Note that we have an infinite sequence of Title variables: if we had a finite number, our BN would not define probabilities for worlds with more than that number of publications. We stipulate that if a basic variable has an object o as an argument, then in worlds where o does not exist, the variable takes on the special value null. Thus, #Pub is a parent of each $\mathsf{Title}(p)$ variable, determining whether that variable takes the value null or not. The set of publications available for selection in the dependency

$$\#\mathsf{Researcher} \sim \mathsf{NumResearchersPrior}()$$

$$\mathsf{Position}(r) \sim [0.7 : \ \mathsf{GradStudent}, 0.2 : \ \mathsf{PostDoc}, 0.1 : \ \mathsf{Prof}]$$

$$\#\mathsf{Pub}(\mathsf{FirstAuthor} = r) \sim \mathsf{NumPubsCPD}(\mathsf{Position}(r))$$

Figure 5. Dependency statements that augment our bibliography model to represent a set of researchers, the position of each researcher, and the set of first-authored publications by each researcher.

statement for $\mathsf{PubCited}(c)$ also depends on the number variable.

Objects of a given type may be generated by more than one event in the generative process. For instance, if we include objects of type Researcher in our model and add a function $\mathsf{FirstAuthor}(p)$ that maps publications to researchers, we may wish to say that each researcher independently generates a crop of papers on which he or she is the first author. The number of papers generated may depend on the researcher's position (graduate student, professor, etc.). We now get a family of number variables $\#\mathsf{Pub}(\mathsf{FirstAuthor} = r)$, where r ranges over researchers. The number of researchers may itself be governed by a number variable. Figure 5 shows the dependency statements for these aspects of the scenario.

In this model, $\mathsf{FirstAuthor}(p)$ is an *origin function*: in the generative model underlying the OUPM, it is set when p is created, not in a separate generative step. The values of origin functions on an object tell us which number variable governs that object's existence; for example, if $\mathsf{FirstAuthor}(p)$ is $\langle \mathit{Researcher}, 5 \rangle$, then $\#\mathsf{Pub}(\mathsf{FirstAuthor} = \langle \mathit{Researcher}, 5 \rangle)$ governs the existence of p. Origin functions can also be used in dependency statements, just like any other function: for instance, we might change the dependency statement for $\mathsf{PubCited}(c)$ so that more significant publications are more likely to be cited, with the significance of a publication p being influenced by $\mathsf{Position}(\mathsf{FirstAuthor}(p))$.

In the scenario we have considered so far, each possible world contains finitely many Researcher and Pub objects. OUPMs can also accommodate infinite numbers of objects. For instance, we could define a model for academia where each researcher r generates a random number of new researchers r' such that $\mathsf{Advisor}(r') = r$. Some possible worlds in this model may contain infinitely many researchers.

3.2 Possible worlds and basic random variables

In defining the semantics of RPMs, we said that a model M defines a BN over its basic random variables $\mathbf{V}_M$, and then we exploited the one-to-one correspondence between full instantiations of those variables and possible worlds. In an OUPM, however, there may be instantiations of the basic random variables that do not correspond to any possible world. An example in our bibliography scenario is an

instantiation where $\#\mathsf{Pub} = 100$, but $\mathsf{Title}(p)$ takes on a non-null value for 200 publications.

To facilitate using the basic variables to define a probability measure over the possible worlds, we would like to have a one-to-one mapping between Ω_M and a set of *achievable* instantiations of $\mathbf{V}_M$. This is straightforward in cases like our first OUPM, where there is only one number variable for each type of object. Then our semantics specifies that the *non-guaranteed objects* of each type—that is, the objects that exist in some possible worlds and not others, like the publications in our example—are pairs $\langle Pub, 1\rangle, \langle Pub, 2\rangle, \ldots$. In each world, the set of non-guaranteed objects of each type that exist is required to be a prefix of this numbered sequence. Thus, if we know that $\#\mathsf{Pub} = 4$ in a world ω, we know that the publications in ω are $\langle Pub, 1\rangle$ through $\langle Pub, 4\rangle$, not some other set of non-guaranteed objects.

Things are more complicated when we have multiple number variables for a type, as in our example with researchers generating publications. Given values for all the number variables of the form $\#\mathsf{Pub}(\mathsf{FirstAuthor} = r)$, we do not want there to be any uncertainty about *which* non-guaranteed objects have each FirstAuthor value. We can achieve this by letting the non-guaranteed objects be nested tuples that encode their generation history. For the publications with $\langle Researcher, 5\rangle$ as their first author, we use tuples

$$\langle Pub, \langle FirstAuthor, \langle Researcher, 5\rangle\rangle, 1\rangle$$
$$\langle Pub, \langle FirstAuthor, \langle Researcher, 5\rangle\rangle, 2\rangle$$

and so on. As before, in each possible world, the set of tuples in each sequence must form a prefix of the sequence. This construction yields the following lemma.

LEMMA 1. *In any OUPM M, each complete instantiation of $\mathbf{V}_M$ is consistent with at most one possible world in Ω_M.*

Section 4.3 of Milch [2006] gives a more rigorous formulation and proof of this result. Given this lemma, the probability measure defined by an OUPM M on Ω_M is well-defined if the OUPM specifies a joint probability distribution for $\mathbf{V}_M$ that is concentrated on the set of achievable instantiations. Since the OUPM's CPDs implicitly force a variable to take the value null when any of its arguments do not exist, any distribution consistent with the CPDs will indeed put probability one on achievable instantiations.

Informally, the probability distribution for the basic random variables can be defined by a generative process that builds up an instantiation step-by-step, sampling a value for each variable according to its dependency statement. In the next section, we show how this intuitive semantics can be formalized using an extended version of Bayesian networks.

4 Extending BN semantics

There are two equivalent ways of defining the probability distribution represented by a BN $\mathbf{B}$. The first is based on conditional independence statements; specifically,

$$
\begin{array}{l}
\#\mathsf{Pub} \sim \mathsf{NumPubsPrior}() \\
\\
\mathsf{Title}(p) \sim \mathsf{TitlePrior}() \\
\\
\mathsf{Date}(c) \sim \mathsf{DatePrior}() \\
\\
\mathsf{SourceCopied}(c) \sim [0.9 : \ \mathsf{null}, \\
\qquad\qquad\qquad\qquad 0.1 : \ \mathsf{Uniform}(\{\mathsf{Citation}\ c_2 : \\
\qquad\qquad\qquad\qquad\qquad (\mathsf{PubCited}(c_2) = \mathsf{PubCited}(c)) \\
\qquad\qquad\qquad\qquad\qquad \wedge\, (\mathsf{Date}(c_2) < \mathsf{Date}(c))\})] \\
\\
\mathsf{CitTitle}(c) \sim \textbf{if}\ \mathsf{SourceCopied}(c) = \mathsf{null} \\
\qquad\qquad\qquad\qquad \textbf{then}\ \mathsf{TitleEditCPD}(\mathsf{Title}(\mathsf{PubCited}(c))) \\
\qquad\qquad\qquad\quad \textbf{else}\ \mathsf{TitleEditCPD}(\mathsf{CitTitle}(\mathsf{SourceCopied}(c)))
\end{array}
$$

Figure 6. Dependency statements for a model where each citation was written on some date, and a citation may copy the title from an earlier citation of the same publication rather than copying the publication title directly.

the directed local Markov property: each variable is conditionally independent of its non-descendants given its parents. The second is based on a product expression for the joint distribution; if σ is any instantiation of the full set of variables $\mathbf{V_B}$ in the BN, then

$$P(\sigma) = \prod_{X \in \mathbf{V_B}} \varphi_X \left(\sigma[X], \sigma\left[\mathrm{Pa}(X)\right]\right) \ .$$

The remarkable property of BNs is that if the graph is finite and acyclic, then there is guaranteed to be exactly one joint distribution that satisfies these conditions.

4.1 Infinite sets of variables

Note that in the BN in Figure 4, the $\mathsf{CitTitle}(c)$ variables have infinitely many parents. The fact that the BN has infinitely many nodes means that we can no longer use the standard product-expression semantics for the BN, because the product of the CPDs for all variables is an infinite product, and will typically be zero for all values of the variables. We would like to specify probabilities for certain partial, finite instantiations of the variables that are sufficient to define the joint distribution. As noted by Kersting and DeRaedt [2001], if it is possible to number the nodes of the BN in topological order, then it suffices to specify the product expression for each finite prefix of this numbering. However, if a variable has infinitely many parents, then the BN has no topological numbering—if we try numbering the nodes in topological order, we will spend forever on X's parents and never reach X.

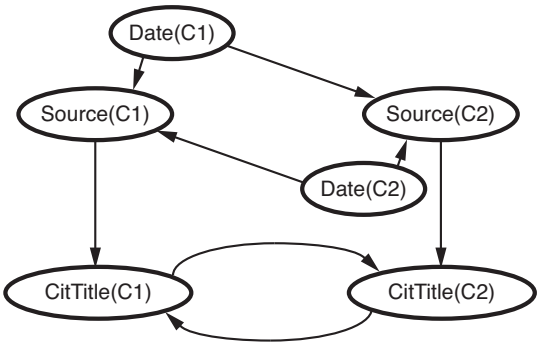


Figure 7. Part of the BN defined by the OUPM in Figure 6, for two citations.

4.2 Cyclic sets of potential dependencies

In OUPMs and even RPMs with relational uncertainty, it is fairly easy to write dependency statements that define a cyclic BN. For instance, suppose that some citations are composed by copying another citation, and we do not know who copied whom. We can specify a model where each citation was written at some unknown date, and with probability 0.1, a citation copies an earlier citation to the same publication if one exists. Figure 6 shows the dependency statements for this model. (Note that Date here is the date the citation was written, i.e., the date of the citing paper, not the date of the paper being cited.)

The BN defined by this OUPM is cyclic, as shown in Figure 7. In general, a cyclic BN may fail to define a distribution; there may be no joint distribution with the specified CPDs. However, in this case, it is intuitively clear that that cannot happen. Since a citation can only copy another citation with a strictly earlier date, the dependencies that are active in any positive-probability world must be acyclic. There are actually elements of the possible world set Ω_M where the dependencies are cyclic: these are worlds where, for some citation c, SourceCopied(c) does not have an earlier date than c. But the CPD for SourceCopied forces these worlds to have probability zero.

The difficult aspect of semantics for this class of cyclic BNs is the directed local Markov property. It is no longer sufficient to assert that X is independent of its non-descendants in the full BN given its parents, because its set of non-descendants in the full BN may be too small. In this model, all the CitTitle nodes are descendants of each other, so the standard directed local Markov property would yield no assertions of conditional independence between them.

4.3 Partition-based semantics for OUPMs

We can solve these difficulties by exploiting the context-specific nature of dependencies in an OUPM, as revealed by dependency statements.[1] For each basic random variable X, an OUPM defines a partition Λ_X of Ω_M. Two worlds are in the same block of this partition if evaluating the dependency statement for X in these two worlds yields the same conditional distribution for X. For instance, in our OUPM for the bibliography domain, the partition blocks for $\mathsf{CitTitle}(\mathsf{Cit1})$ are sets of worlds that agree on the value of $\mathsf{Title}(\mathsf{PubCited}(\mathsf{Cit1}))$. For each block $\lambda \in \Lambda_X$, the OUPM defines a probability distribution $\varphi_X(x, \lambda)$ over values of X.

One defining property of the probability measure P_M specified by an OUPM M is that for each basic random variable $X \in \mathbf{V}_M$ and each partition block $\lambda \in \Lambda_X$,

$$P_M(X = x \mid \lambda) = \varphi_X(x, \lambda) \tag{1}$$

To fully define P_M, however, we need to make an assertion analogous to a BN's factorization property or directed local Markov property. We will say that a partial instantiation σ *supports* a random variable X if there is some block $\lambda \in \Lambda_X$ such that $\sigma \subseteq \lambda$. An instantiation that supports X in an OUPM is an analogous to an instantiation that assigns values to all the parents of X in a BN. We define an instantiation σ to be *self-supporting* if for each variable $X \in \text{vars}(\sigma)$, the restriction of σ to $\text{vars}(\sigma) \setminus \{X\}$ (denoted σ_{-X}) supports X. We can now state a factorization property for OUPMs.

PROPERTY 2 (Factorization property for an OUPM M). For each finite, self-supporting instantiation σ on $\mathbf{V}_M$,

$$P_M(\sigma) = \prod_{X \in \text{vars}(\sigma)} \varphi_X(\sigma[X], \lambda_X(\sigma_{-X}))$$

where $\lambda_X(\sigma_{-X})$ is the partition block in Λ_X that has σ_{-X} as a subset.

We can also define an analogue of the directed local Markov property for OUPMs. Recall that in the BN case, the directed local Markov property asserts that X is conditionally independent of every subset of its non-descendants given $\text{Pa}(X)$. In fact, it turns out to be sufficient to make this assertion for only a special class of non-descendant subsets, namely those that are *ancestral* (closed under the parent relation). Any ancestral set of variables that does not contain X contains only non-descendants of X. So in the BN case, we can reformulate the directed local Markov property to assert that given $\text{Pa}(X)$, X is conditionally independent of any ancestral set of variables that does not contain X.

In OUPMs, the equivalent of a variable set that is closed under the parent relation is a self-supporting instantiation. We can formulate the directed local Markov property for an OUPM M as follows:

[1] We will assume all random variables are discrete in this treatment, but the ideas can be extended to the continuous case.

PROPERTY 3 (Directed local Markov property for an OUPM M). For each basic random variable $X \in \mathbf{V}_M$, each block $\lambda \in \Lambda_X$, and each self-supporting instantiation σ on $\mathbf{V}_M$ such that $X \notin \text{vars}(\sigma)$, X is conditionally independent of σ given λ.

Under what conditions is there a unique probability measure P_M on Ω_M that satisfies Properties 2 and 3? In the BN case, it suffices for the graph to admit a topological numbering. We can define a similar notion that is specific to individual worlds: a *supportive numbering* for a world $\omega \in \Omega_M$ is a numbering $X_0, X_1, \ldots$ of $\mathbf{V}_M$ such that for each natural number n, the instantiation $(X_0(\omega), \ldots, X_{n-1}(\omega))$ supports X_n.

THEOREM 4. *Let M be an OUPM such that for every world $\omega \in \Omega_M$, either:*

- *ω has a supportive numbering, or*
- *for some basic random variable $X \in \mathbf{V}_M$, $\varphi_X\,(X(\omega),\, \lambda_X(\omega)) = 0$.*

Then there is exactly one probability measure on Ω_M satisfying the factorization property (Property 2), and it is also the unique probability measure that satisfies both Equation 1 and the directed local Markov property (Property 3).

This theorem follows from Lemma 1 and results proved in Section 3.4 of Milch [2006]. Note that the theorem does not require supportive numberings for worlds that are *directly disallowed*—that is, those that are forced to have probability zero by the CPD for some variable.

In our basic bibliography scenario with unknown publications, we can construct a supportive numbering for each possible world ω by taking first the number variable #Pub, then the $\mathsf{PubCited}(c)$ variables, then the $\mathsf{Title}(p)$ variables for the publications that serve as values of $\mathsf{PubCited}(c)$ variables in ω, then the $\mathsf{CitTitle}(c)$ variables, and finally the infinitely many $\mathsf{Title}(p)$ variables for publications that are uncited or do not exist in ω. For the scenario where citation titles can be copied from earlier citations, we have to add the $\mathsf{Date}(c)$ variables and then the $\mathsf{SourceCopied}(c)$ variables before the $\mathsf{CitTitle}(c)$ variables. We order the $\mathsf{CitTitle}(c)$ variables in a way that is consistent with $\mathsf{Date}(c)$. This procedure yields a supportive numbering in all worlds except those where $\exists c\ \mathsf{Date}(\mathsf{SourceCopied}(c)) \geq \mathsf{Date}(c)$, but such worlds are directly disallowed by the CPD for $\mathsf{SourceCopied}(c)$.

4.4 Representing OUPMs as contingent Bayesian networks

The semantics we have given for OUPMs so far does not make reference to any graph. But we can also view an OUPM as defining a *contingent Bayesian network* (CBN) [Milch et al. 2005], which is a BN where each edge is labeled with an event. The event indicates when the edge is active, in a sense we will soon make precise. Figures 8 and 9 show CBNs corresponding to the infinite BN in Figure 4 and the cyclic BN in Figure 7, respectively.

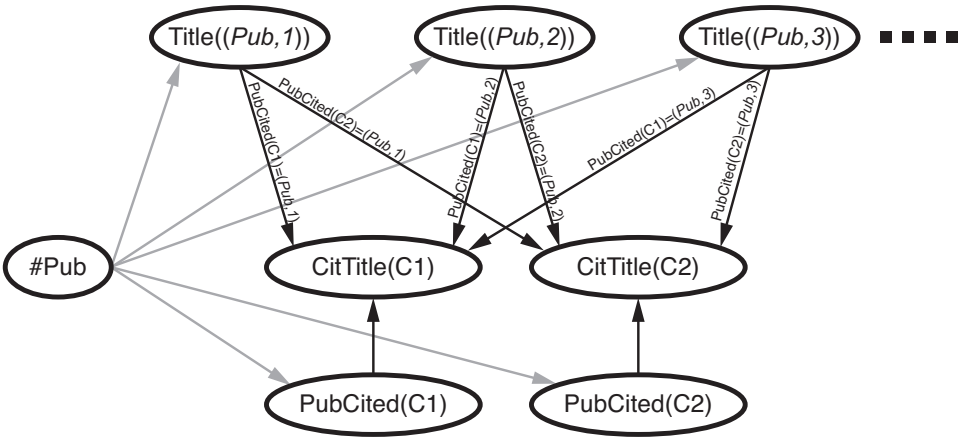


Figure 8. A contingent BN for the bibliography scenario with unknown objects.

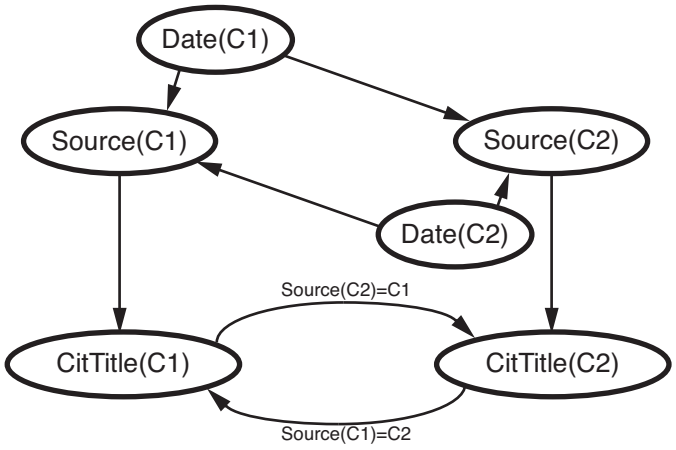


Figure 9. Part of a contingent BN for the OUPM in Figure 6.

A CBN can be viewed as a partition-based model where the partition Λ_X for each random variable X is defined by a decision tree. The internal nodes in this decision tree are labeled with random variables; the edges are labeled with variable values; and the leaves specify conditional probability distributions for X. The blocks in Λ_X correspond to the leaves in this tree (we assume the tree has no infinite paths, so the leaves cover all possible worlds). The restriction to decision trees allows us to define a notion of a parent being active in a particular world: if we walk along X's tree from the root, following edges consistent with a given world ω, then the random variables on the nodes we visit are the active parents of X in ω. The label on an edge $W \rightarrow X$ in a CBN is the event consisting of those worlds where W is an active parent of X. (In diagrams, we omit the trivial label $A = \Omega_M$, which indicates that the dependency is always active.)

The abstract notions of a self-supporting instantiation and a supportive numbering have simple graphical analogues in a CBN. We will use $\mathbf{B}^\sigma$ to denote the BN obtained from a CBN $\mathbf{B}$ by keeping only those edges whose conditions are entailed by σ. An instantiation σ supports a variable X if and only if all the parents of X in $\mathbf{B}^\sigma$ are in vars(σ), and it is self-supporting if and only if vars(σ) is an ancestral set in $\mathbf{B}^\sigma$. A supportive numbering for a world ω is a topological numbering of the BN $\mathbf{B}^\omega$ obtained by keeping only those edges whose conditions are satisfied by ω. Thus, the well-definedness condition in Theorem 4 can be stated for CBNs as follows: for each world $\omega \in \Omega_M$ that is not directly disallowed, $\mathbf{B}^\omega$ must have a topological numbering.

Not all partitions can be represented exactly as the leaves of a decision tree, so there are sets of context-specific independence properties that can be captured by OUPMs and not CBNs. However, when we perform inference on an OUPM, we typically use a function that evaluates the dependency statement for each variable, looking up the values of other random variables in a given world (or partial instantiation) as needed. For example, a function evaluating the dependency statement for CitTitle(Cit1) will always access PubCited(Cit1), and then it will access a particular Title variable depending on the value of the PubCited variable. This evaluation process implicitly defines a decision tree; the order of splits in the tree depends on the evaluation order used. When we discuss inference for OUPMs, we will assume that we are operating on the CBN implicitly defined by some evaluation function.

5 Inference

Given an OUPM, we would like to be able to compute the probability of a query event Q given an evidence event E. For example, Q could be the event that PubCited(Cit1) = PubCited(Cit2) and E could be the event that CitTitle(Cit1) = "Learning Probabilistic Relational Models" and CitTitle(Cit2) = "Learning Probabilitsic Relation Models". The ideas we present can be extended to other tasks such as computing the posterior distribution of a random variable, or finding the *maximum a posteriori* (MAP) assignment of values to a set of random variables.

5.1 MCMC over partial worlds

Sampling-based or Monte Carlo inference algorithms are well-suited for OUPMs because each sample specifies what objects exist and what relations hold among them. We focus on Markov chain Monte Carlo (MCMC), where we simulate a Markov chain over possible worlds consistent with the evidence E, such that the stationary distribution of the chain is the posterior distribution over worlds given E. Such a chain can be constructed using the Metropolis–Hastings method, where we use an arbitrary *proposal distribution* $q(\omega'|\omega_t)$, but accept or reject each proposal based on the relative probabilities of ω' and ω_t.

Specifically, at each step t in our Markov chain, we sample ω' from $q(\omega'|\omega_t)$ and

then compute the *acceptance probability*:

$$\alpha = \min\left(1, \frac{P_M(\omega')q(\omega_t|\omega')}{P_M(\omega_t)q(\omega'|\omega_t)}\right) .$$

With probability α, we accept the proposal and let $\omega_{t+1} = \omega'$; otherwise we reject the proposal and let $\omega_{t+1} = \omega_t$.

The difficulty in OUPMs is that each world may be very large. For instance, if we have a world where $\#Pub = 1000$, but only 100 publications are referred to by our observed citations, then the world must also specify the titles of the 900 unobserved publications. Sampling values for these 900 Title variables and computing their probabilities will slow down our algorithm unnecessarily. In scenarios where some possible worlds have infinitely many objects, specifying a possible world completely may be impossible.

Thus, we would like to run MCMC over *partial* descriptions that specify values only for certain random variables. The set of instantiated variables may vary from world to world. Since a partial instantiation σ defines an event (the set of worlds that are consistent with it), a Markov chain over partial instantiations can be viewed as a chain over events. Thus, we use the acceptance probability:

$$\alpha = \min\left(1, \frac{P_M(\sigma')q(\sigma_t|\sigma')}{P_M(\sigma_t)q(\sigma'|\sigma_t)}\right)$$

where $P_M(\sigma)$ is the probability of the event σ. As long as the set Σ of partial instantiations that can be returned by q forms a partition of E, and each partial instantiation is specific enough to determine whether Q is true, we can estimate $P(Q|E)$ using a Markov chain on Σ with stationary distribution proportional to $P_M(\sigma)$ [Milch and Russell 2006].

In general, computing the probability $P_M(\sigma)$ involves summing over all the variables not instantiated in σ—which is precisely what we want to avoid by using a Monte Carlo inference algorithm. Fortunately, if each instantiation in Σ is self-supporting, we can compute its probability using the product expression from Property 2. Thus, our partial worlds are self-supporting instantiations that include the query and evidence variables. We also make sure to use *minimal* instantiations satisfying this condition—that is, instantiations that would cease to be self-supporting if we removed any non-query, non-evidence variable. It can be shown that in a CBN, such minimal self-supporting instantiations are mutually exclusive . So if our set of partial worlds Σ covers all of E, we are guaranteed to have a partition of E, as required. An example of a partial world in our bibliography scenario is:

$\#\mathsf{Pub} = 50, \mathsf{CitTitle}(\mathsf{Cit1}) =$ "Calculus", $\mathsf{CitTitle}(\mathsf{Cit2}) =$ "Intro to Calculus",
$\mathsf{PubCited}(\mathsf{Cit1}) = \langle Pub, 17\rangle, \mathsf{PubCited}(\mathsf{Cit2}) = \langle Pub, 31\rangle$,
$\mathsf{Title}(\langle Pub, 17\rangle) =$ "Calculus", $\mathsf{Title}(\langle Pub, 31\rangle) =$ "Intro to Calculus"

5.2 Abstract partial worlds

In the partial instantiation above, we specify the tuple representation of each publication, as in $\mathsf{PubCited}(\mathsf{Cit1}) = \langle Pub, 17 \rangle$. If partial worlds are represented this way, then the code that implements the proposal distribution has to choose numbers for any new objects it adds, keep track of the probability of its choices, and compute the probability of the reverse proposal. Some kinds of moves are impossible unless the proposer renumbers the objects: for instance, the total number of publications cannot be decreased from 1000 to 900 when publication 941 is in use.

To simplify the proposal distribution, we can use partial worlds that abstract away the identities of objects using existential quantifiers:

$\exists$ distinct x, y
$\#\mathsf{Pub} = 50, \mathsf{CitTitle}(\mathsf{Cit1}) =$ "Calculus", $\mathsf{CitTitle}(\mathsf{Cit2}) =$ "Intro to Calculus",
$\mathsf{PubCited}(\mathsf{Cit1}) = x, \mathsf{PubCited}(\mathsf{Cit2}) = y,$
$\mathsf{Title}(x) =$ "Calculus", $\mathsf{Title}(y) =$ "Intro to Calculus"

The probability of the event corresponding to an abstract partial world depends on the number of ways the logical variables can be mapped to distinct objects. For simplicity, we will assume that there is only one number variable for each type. If an abstract partial world σ uses logical variables for a type τ, we require it to instantiate the number variable for that type. We also require that for each logical variable x, there is a distinct ground term t_x such that σ implies $t_x = x$; this ensures that each mapping from logical variables to tuple representations yields a distinct possible world. Let T be the set of types of logical variables in σ, and for each type $\tau \in T$, let n_τ be the value of $\#\tau$ in σ and ℓ_τ be the number of logical variables of type τ in σ. Then we have:

$$P(\sigma) = P_c(\sigma) \prod_{\tau \in T} \frac{n_\tau!}{(n_\tau - \ell_\tau)!}$$

where $P_c(\sigma)$ is the probability of any one of the "concrete" instantiations obtained by substituting distinct tuple representations for the logical variables in σ.

5.3 Locality of computation

Given a current instantiation σ_t and a proposed instantiation σ', computing the acceptance probability involves computing the ratio:

$$\frac{P_M(\sigma')q(\sigma_t|\sigma')}{P_M(\sigma_t)q(\sigma'|\sigma_t)} = \frac{q(\sigma_t|\sigma') \prod_{X \in \mathrm{vars}(\sigma')} \varphi_X\left(\sigma'[X], \sigma'\left[\mathrm{Pa}_{\sigma'}(X)\right]\right)}{q(\sigma'|\sigma_t) \prod_{X \in \mathrm{vars}(\sigma_t)} \varphi_X\left(\sigma_t[X], \sigma_t\left[\mathrm{Pa}_{\sigma_t}(X)\right]\right)}$$

where $\mathrm{Pa}_\sigma(X)$ is the set of parents of X whose edge conditions are entailed by σ. This expression is daunting, because even though the instantiations σ_t and σ' are only partial descriptions of possible worlds, they may still assign values to large sets of random variables — and the number of instantiated variables grows at least linearly with the number of observations we have. Since we may want to run

millions of MCMC steps, having each step take time proportional to the number of observations would make inference prohibitively expensive.

Fortunately, with most proposal distributions used in practice, each step changes the values of only a small set of random variables. Furthermore, if the edges that are active in any given possible world are fairly sparse, then $\sigma\,[\mathrm{Pa}_{\sigma'}(X)]$ will also be the same as $\sigma_t\,[\mathrm{Pa}_{\sigma_t}(X)]$ for many variables X. Thus, many factors will cancel out in the ratio above.

We need to compute the "new" and "old" probability factors for a variable X only if either $\sigma'[X] \neq \sigma_t[X]$, or there is some active parent $W \in \mathrm{Pa}_{\sigma_t}(X)$ such that $\sigma'[W] \neq \sigma_t[W]$. (We take these inequalities to include the case where σ' assigns a value to the variable and σ_t does not, or vice versa.) Note that it is not possible for $\mathrm{Pa}_{\sigma'}(X)$ to be different from $\mathrm{Pa}_{\sigma_t}(X)$ unless one of the "old" active parents in $\mathrm{Pa}_{\sigma_t}(X)$ has changed: given that σ_t is a self-supporting instantiation, the values of X's instantiated parents in σ_t determine the truth values of the conditions on all the edges into X, so the set of active edges into X cannot change unless one of these parent variables changes.

This fact is exploited in the BLOG system [Milch and Russell 2006] to efficiently detect which probability factors need to be computed for a given proposal. The system maintains a graph of the edges that are active in the current instantiation σ_t. The proposer provides a list of the variables that are changed in σ', and the system follows the active edges in the graph to identify the children of these variables, whose probability factors also need to be computed. Thus, the graphical locality that is central to many other BN inference algorithms also plays a role in MCMC over relational structures.

6 Related work

The connection between probability and first-order languages was first studied by Carnap [1950]. Gaifman [1964] and Scott and Krauss [1966] defined a formal semantics whereby probabilities could be associated with first-order sentences and for which models were probability measures on possible worlds. Within AI, this idea was developed for propositional logic by Nilsson [1986] and for first-order logic by Halpern [1990]. The basic idea is that each sentence *constrains* the distribution over possible worlds; one sentence entails another if it expresses a stronger constraint. For example, the sentence $\forall x\, P(\mathsf{Hungry}(x)) > 0.2$ rules out distributions in which any object is hungry with probability less than 0.2; thus, it entails the sentence $\forall x\, P(\mathsf{Hungry}(x)) > 0.1$. Bacchus [1990] investigated knowledge representation issues in such languages. It turns out that writing a *consistent* set of sentences in these languages is quite difficult and constructing a unique probability model nearly impossible unless one adopts the representational approach of Bayesian networks by writing suitable sentences about conditional probabilities.

The impetus for the next phase of work came from researchers working with BNs directly. Rather than laboriously constructing large BNs by hand, they built

them by composing and instantiating "templates" with logical variables that described local causal models associated with objects [Breese 1992; Wellman et al. 1992]. The most important such language was BUGS (Bayesian inference Using Gibbs Sampling) [Gilks et al. 1994], which combined Bayesian networks with the indexed-random-variable notation common in statistics. These languages inherited the key property of Bayesian networks: every well-formed knowledge base defines a unique, consistent probability model. Languages with well-defined semantics based on unique names and domain closure drew on the representational capabilities of logic programming [Poole 1993; Sato and Kameya 1997; Kersting and De Raedt 2001] and semantic networks [Koller and Pfeffer 1998; Pfeffer 2000]. Initially, inference in these models was performed on the equivalent Bayesian network. *Lifted* inference techniques borrow from first-order logic the idea of performing an inference once to cover an entire equivalence class of objects [Poole 2003; de Salvo Braz et al. 2007; Kisynski and Poole 2009]. MCMC over relational structures was introduced by Pasula and Russell [2001]. Getoor and Taskar [2007] collect many important papers on first-order probability models and their use in machine learning.

Probabilistic reasoning about identity uncertainty has two distinct origins. In statistics, the problem of *record linkage* arises when data records do not contain standard unique identifiers—for example, in financial, medical, census, and other data [Dunn 1946; Fellegi and Sunter 1969]. In control theory, the problem of *data association* arises in multitarget tracking when each detected signal does not identify the object that generated it [Sittler 1964]. For most of its history, work in symbolic AI assumed erroneously that sensors could supply sentences with unique identifiers for objects. The issue was studied in the context of language understanding by Charniak and Goldman [1993] and in the context of surveillance by Huang and Russell [1998] and Pasula *et al.* [1999]. Pasula *et al.* [2003] developed a complex generative model for authors, papers, and citation strings, involving both relational and identity uncertainty, and demonstrated high accuracy for citation information extraction. The first formally defined language for open-universe probability models was BLOG [Milch et al. 2005], from which the material in the current chapter was developed. Laskey [2008] describes another open-universe modeling language called multi-entity Bayesian networks.

Another important thread goes under the name of *probabilistic programming languages*, which include IBAL [Pfeffer 2007] and CHURCH [Goodman et al. 2008]. These languages represent first-order probability models using a programming language extended with a randomization primitive; any given "run" of a program can be seen as constructing a possible world, and the probability of that world is the probability of all runs that construct it.

The OUPMs we have described here bear some resemblance to probabilistic programs, since each dependency statement can be viewed as a program fragment for sampling a value for a child variable. However, expressions in dependency statements have different semantics from those in a probabilistic functional language

such as IBAL: if an expression such as Title(Pub5) is evaluated in several dependency statements in a given possible world, it returns the same value every time, whereas the value of an expression in IBAL is sampled independently each time it appears. The CHURCH language incorporates aspects of both approaches: it includes a *stochastic memoization* construct that lets the programmer designate certain expressions as having values that are sampled once and then reused. McAllester *et al.* [2008] define a probabilistic programming language that makes sources of random bits explicit and has a possible-worlds semantics similar to OUPMs.

This chapter has described generative, directed models. The combination of relational and first-order notations with (undirected) Markov networks is also interesting [Taskar et al. 2002; Richardson and Domingos 2006]. Undirected formalisms are convenient because there is no need to avoid cycles. On the other hand, an essential assumption underlying relational probability models is that one set of CPD parameters is appropriate for a wide range of relational structures. For instance, in our RPMs, the prior for a publication's title does not depend on how many citations refer to it. But in an undirected model, adding more citations to a publication (and thus more potentials linking Title(p) to CitTitle(c) variables) will usually change the marginal on Title(p), even when none of the CitTitle(c) values are observed. This suggests that all the potentials must be learned jointly on a training set with roughly the same distribution of relational structures as the test set; in the directed case, we are free to learn different CPDs from different data sources.

7 Discussion

This chapter has stressed the importance of unifying probability theory with first-order logic—particularly for cases with unknown objects—and has presented one possible approach based on open-universe probability models, or OUPMs. OUPMs draw on the key idea introduced into AI by Judea Pearl: generative probability models based on local conditional distributions. Whereas BNs generate worlds by assigning values to variables one at a time, relational models can assign values to a whole class of variables through a single dependency assertion, while OUPMs add object creation as one of the generative steps.

OUPMs appear to enable the straightforward representation of a wide range of situations. In addition to the citation model mentioned in this chapter (see Milch [2006] for full details), models have been written for multitarget tracking, plan recognition, sibyl attacks (a security threat in which a reputation system is compromised by individuals who create many fake identities), and detection of nuclear explosions using networks of seismic sensors [Russell and Vaidya 2009]. In each case, the model is essentially a transliteration of the obvious English description of the generative process.

Inference, however, is another matter. The generic Metropolis–Hastings inference engine written for BLOG in 2006 is far too slow to support any of the applications described in the preceding paragraph. For the citation problem, Milch [2006] de-

scribes an application-specific proposal distribution for the generic M–H sampler that achieves speeds comparable to a completely hand-coded, application-specific inference engine. This approach is feasible in general but requires a significant coding effort by the user. Current efforts in the BLOG project are aimed instead at improving the generic engine: implementing a generalized Gibbs sampler for structurally varying models; enabling the user to specify blocks of variables that are to be sampled jointly to avoid problems with slow mixing; borrowing compiler techniques from the logic programming field to reduce the constant factors; and building in parametric learning. With these changes, we expect BLOG to be usable over a wide range of applications with only minimal user intervention.

References

Bacchus, F. (1990). *Representing and Reasoning with Probabilistic Knowledge*. MIT Press.

Breese, J. S. (1992). Construction of belief and decision networks. *Computational Intelligence 8*(4), 624–647.

Carnap, R. (1950). *Logical Foundations of Probability*. Univ. of Chicago Press.

Charniak, E. and R. P. Goldman (1993). A Bayesian model of plan recognition. *Artificial Intelligence 64*(1), 53–79.

de Salvo Braz, R., E. Amir, and D. Roth (2007). Lifted first-order probabilistic inference. In L. Getoor and B. Taskar (Eds.), *Introduction to Statistical Relational Learning*. MIT Press.

Dunn, H. L. (1946). Record linkage. *Am. J. Public Health 36*(12), 1412–1416.

Fellegi, I. and A. Sunter (1969). A theory for record linkage. *J. Amer. Stat. Assoc. 64*, 1183–1210.

Gaifman, H. (1964). Concerning measures in first order calculi. *Israel J. Math. 2*, 1–18.

Getoor, L. and B. Taskar (Eds.) (2007). *Introduction to Statistical Relational Learning*. MIT Press.

Gilks, W. R., A. Thomas, and D. J. Spiegelhalter (1994). A language and program for complex Bayesian modelling. *The Statistician 43*(1), 169–177.

Goodman, N. D., V. K. Mansinghka, D. Roy, K. Bonawitz, and J. B. Tenenbaum (2008). Church: A language for generative models. In *Proc. 24th Conf. on Uncertainty in AI*.

Halpern, J. Y. (1990). An analysis of first-order logics of probability. *Artificial Intelligence 46*, 311–350.

Huang, T. and S. J. Russell (1998). Object identification: A Bayesian analysis with application to traffic surveillance. *Artificial Intelligence 103*, 1–17.

Kersting, K. and L. De Raedt (2001). Adaptive Bayesian logic programs. In *Proc. 11th International Conf. on Inductive Logic Programming*, pp. 104–117.

Kisynski, J. and D. Poole (2009). Lifted aggregation in directed first-order probabilistic models. In *Proc. 21st International Joint Conf. on Artificial Intelligence*, pp. 1922–1929.

Koller, D. and A. Pfeffer (1998). Probabilistic frame-based systems. In *Proc. 15th AAAI National Conf. on Artificial Intelligence*, pp. 580–587.

Laskey, K. B. (2008). MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence 172*, 140–178.

McAllester, D., B. Milch, and N. D. Goodman (2008). Random-world semantics and syntactic independence for expressive languages. Technical Report MIT-CSAIL-TR-2008-025, Massachusetts Institute of Technology.

Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov (2005). BLOG: Probabilistic models with unknown objects. In *Proc. 19th International Joint Conf. on Artificial Intelligence*, pp. 1352–1359.

Milch, B., B. Marthi, D. Sontag, S. Russell, D. L. Ong, and A. Kolobov (2005). Approximate inference for infinite contingent Bayesian networks. In *Proc. 10th International Workshop on Artificial Intelligence and Statistics*.

Milch, B. and S. Russell (2006). General-purpose MCMC inference over relational structures. In *Proc. 22nd Conf. on Uncertainty in Artificial Intelligence*, pp. 349–358.

Milch, B. C. (2006). *Probabilistic Models with Unknown Objects*. Ph.D. thesis, Univ. of California, Berkeley.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence 28*(1), 71–87.

Pasula, H., B. Marthi, B. Milch, S. Russell, and I. Shpitser (2003). Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems 15*. MIT Press.

Pasula, H. and S. Russell (2001). Approximate inference for first-order probabilistic languages. In *Proc. 17th International Joint Conf. on Artificial Intelligence*, pp. 741–748.

Pasula, H., S. J. Russell, M. Ostland, and Y. Ritov (1999). Tracking many objects with many sensors. In *Proc. 16th International Joint Conf. on Artificial Intelligence*, pp. 1160–1171.

Pfeffer, A. (2000). *Probabilistic Reasoning for Complex Systems*. Ph.D. thesis, Stanford Univ.

Pfeffer, A. (2007). The design and implementation of IBAL: A general-purpose probabilistic language. In L. Getoor and B. Taskar (Eds.), *Introduction to Statistical Relational Learning*. MIT Press.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence 64*(1), 81–129.

Poole, D. (2003). First-order probabilistic inference. In *Proc. 18th International Joint Conf. on Artificial Intelligence*, pp. 985–991.

Richardson, M. and P. Domingos (2006). Markov logic networks. *Machine Learning 62*, 107–136.

Russell, S. and S. Vaidya (2009). Machine learning and data mining for Comprehensive Test Ban Treaty monitoring. Technical Report LLNL-TR-416780, Lawrence Livermore National Laboratory.

Sato, T. and Y. Kameya (1997). PRISM: A symbolic–statistical modeling language. In *Proc. 15th International Joint Conf. on Artificial Intelligence*, pp. 1330–1335.

Scott, D. and P. Krauss (1966). Assigning probabilities to logical formulas. In J. Hintikka and P. Suppes (Eds.), *Aspects of Inductive Logic*. North-Holland.

Sittler, R. W. (1964). An optimal data association problem in surveillance theory. *IEEE Trans. Military Electronics MIL-8*, 125–139.

Taskar, B., P. Abbeel, and D. Koller (2002). Discriminative probabilistic models for relational data. In *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, pp. 485–492.

Wellman, M. P., J. S. Breese, and R. P. Goldman (1992). From knowledge bases to decision models. *Knowledge Engineering Review 7*, 35–53.

14

A Heuristic Procedure for Finding Hidden Variables

Azaria Paz

1 Introduction

This paper investigates Probabilistic Distribution (PD) induced independency relations which are representable by Directed Acyclic Graphs (DAGs), and are marginalized over a subset of their variables. PD-induced relations have been shown in the literature to be representable as relations that can be defined on various graphical models. All those graphical models have two basic properties: They are *compact*, i.e., the space required for storing such a model is polynomial in the number of variables, and they are *decidable*, i.e., a polynomial algorithm exists for testing whether a given independency is represented in the model. In particular, two such models will be encountered in this paper; the DAG model and the Annotated Graph (AG) model. The reader is supposed to be familiar with the DAG-model which was studied extensively in the literature. An ample introduction to the DAG model is included in Pearl [7, 1988], Pearl [8, 2000], and Lauritzen [2, 1996].[1]

The AG-model in a general form was introduced by Paz, Geva, and Studeny in [5, 2000] and a restricted form of this model, which is all we need for this paper, was introduced by Paz [3, 2003a] and investigated further in Paz [4, 2003b]. For the sake of completeness, we shall reproduce here some of the basic definitions and properties of those models which are relevant for this paper.

Given a DAG-representable PD-induced relation it is often the case that we need to marginalize the relation over a subset of variables. Unfortunately it is seldom the case that such a marginalized relation can be represented by a DAG, which is an easy to manage and a well understood model.

In the paper [4, 2003] the author proved a set of necessary corelations for a given AG to be equivalent to a DAG (see Lemma 1 in the next section). In the same paper a decision procedure is given for checking whether a given AG which satisfies the necessary conditions is equivalent to a DAG. Moreover, if the answer is "yes", the procedure constructs an equivalent DAG to the given AG. In a subsequent paper [6, 2006], the author generalizes the AG model and gives a procedure which enables the representation of any marginalized DAG representable relation by a generalized model.

[1] The main part of this work was done while the author visited the Cognitive Systems Laboratory at UCLA and was supported in part by grants from Air Force, NSF and ONR (MURI).

2 Preliminaries

2.1 Definitions and notations

UGs will denote undirected graphs $G = (V, E)$ where V is a set of *vertices* and E is a set of undirected *edges* connecting between two vertices. Two vertices connected by an edge are *adjacent* or *neighbors*. A path in G of length k is a sequence of vertices $v_1 \dots v_{k+1}$ such that (v_i, v_{i+1}) is an edge in E for $i < 1, \dots, k$. A DAG is an acyclic directed graph $D = (V, E)$ where V is a set of vertices and E is a set of directed *arcs* connecting between two vertices in V. A *trail* of length k in D is a sequence $v_1 \dots v_{k+1}$ of vertices in V such that (v_i, v_{i+1}) is an arc in E for $i = 1 \dots k$. If all the arcs on the trail are directed in the same direction then the trail is called a directed path. If a directed path exists in D from v_i to v_j then v_j is a *descendant* of v_i and v_i is a *predecessor* or *ancestor* of v_j. If the path is of length one then v_i is a *parent* of v_j who is a *child* of v_i.

The *skeleton* of a DAG is the UG derived from the DAG when the orientations of the arcs are removed. A pattern of the form $v_i \rightarrow v_j \leftarrow v_k$ is a *collider pattern* where v_j is the *collider*. If there is no arc between v_i and v_k then v_j is an *uncoupled collider*. The moralizing procedure is the procedure generating a UG from a DAG, by first joining both parents of uncoupled colliders in the DAG by an arc, and then removing the orientation of all arcs. The edges resulting from the coupling of the uncoupled collider are called *moral edges*. As mentioned in the introduction UG's and DAG's represent PD-induced relations whose elements are triplets $t = (X; Y|Z)$ over the set of vertices of the graphs. For a given triplet t we denote by $v(t)$ the set of vertices $v(t) = X \cup Y \cup Z$. Two graph models are *equivalent* if they represent the *same* relation.

2.2 DAG-model

Let $D = (V, E)$ be a DAG whose vertices are V and whose arcs are E. D represents the relation $R(D) = \{t = (X; Y|Z) | t \in D\}$ where X, Y, Z are disjoint subsets of V, the vertices in V represent variables in a PD, t is interpreted as "X is independent of Y given Z" and $t \in D$ means: t is represented in D. To check whether a given triplet t is represented in D we use the Algorithm L1 below due to Lauritzen et al. [1, 1990].

Algorithm L1:
Input: $D = (V, E)$ and $t = (X; Y|Z)$.

1. Let V' be the set of ancestors of $v(t) = X \cup Y \cup Z$ and let $D'(t)$ be the subgraph of D over V'.
2. Moralize $D'(t)$ (i.e., join all uncoupled parents of uncoupled colliders in $D'(t)$). Denote the resulting graph by $D''(t)$.
3. Remove all orientations in $D''(t)$ and denote the resulting UG by $G(D''(t))$.
4. $t \in G(D''(t))$ iff $t \in D$.

REMARK 1. $t \in G$ where G is a UG if and only if Z is a cutset in G (not necessarily minimal) between X and Y.

The definition above and the L1 Algorithm show that the DAG model is both compact and decidable.

2.3 Annotated Graph – model

Let $D = (V, E)$ be a DAG. We derive from D an AG $A = (G, K)$ where G is a UG And K is a set of elements $K = \{e = (d, r(d))\}$ as follows: G is derived from D by moralizing D and removing all orientations from it.

For every moral edge d in G we put an element $e = (d, r(d))$ in K such that $d(a, b)$, the *domain* of e, is the pair of endpoints of the moral edge and $r(d)$, the *range* of e, is the set of vertices including all the uncoupled colliders in D whose parents are a and b, and all the successors of those colliders. Notice that d denotes both a moral edge and the pair of its endpoints. The relation $R(A)$ defined by the AG A is the relation below:

$$R(A) = \{t = (X; Y|Z)|t \in A\}$$

In order to check whether $t \in A$ we use the algorithm L2 due to Paz [3, 2003a] below.

Algorithm L2
Input: An AG $A = (G, K)$.

1. For every element $e = (d, r(d))$ in K such that $r(d) \cap v(t) = \emptyset$ ($v(t) = X \cup Y \cup Z$). Disconnect the edge (a, b) in G corresponding to d and remove from G all the vertices in $r(d)$ and incident edges. Denote the resulting UG by $G(t)$.

2. $t \in A$ if and only if $t \in G(t)$.

REMARK 2. It is clear from the definitions and from the L2 Algorithm that the AG model is both compact and decidable. In addition, it was shown in [3, 2003a] that the AG model has the following uniqueness property: $R(A_1) = R(A_2)$ implies that $A_1 = A_2$ when A_1 and A_2 are AG's. This property does not hold for DAG models where it is possible for two different (and equivalent) DAGs to define the same relation. In fact the AG (D) derived from a DAG D represents the equivalence class of all DAGs which are equivalent to the given DAG D.

REMARK 3. The AGs derived from DAG's are a particular case of AGs as defined in Paz et al. [5, 2000] and there are additional ways to derive AGs that represent PD-induced relations which are not DAG-representable. Consider e.g., the example below. It was shown by Pearl [7, 1988 Ch. 3] that every DAG representable relation is a PD-induced relation. Therefore the relation defined by the DAG in Fig. 1 represents a PD-induced relation.

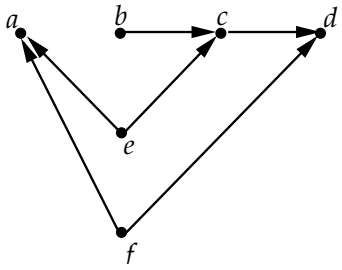


Figure 1. DAG representing relation

If we marginalize this relation over the vertices e and f we get another relation, PD-induced, that can be represented by the AG A in Fig. 2, as will be shown in the sequel, under the semantics of the L2 Algorithm, with $R(A) =$

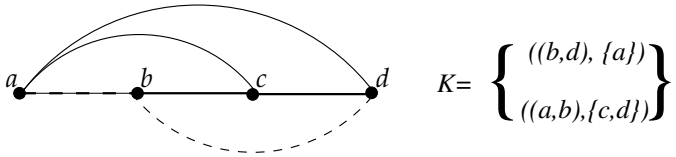


Figure 2. AG A representing a marginalized relation

$\{(a; b|\emptyset), (b; d|c) +$ symmetric images$\}$. But $R(A)$ above cannot be represented by a DAG. This follows from the following lemma that was proven in [4, 2003b].

LEMMA 1. *Let $(G(D), K(D))$ be the annotated graph representation of a DAG D. $K(D)$ has the following properties:*

1. *For every element $((a, b), r) \in K(D)$, there is a vertex $v \in r$ which is a child of both a and b and every vertex $w \in r$ is connected to some vertex v in r whose parents are both a and b.*

2. *For any two elements $(d_1, r_1), (d_2, r_2)$ in $K(D)$, if $d_1 = d_2$ then $r_1 = r_2$.*

3. *For every $((a, b), r) \in K(D)$, (a, b) is an edge in $G(D)$.*

4. *The set of elements $K(D)$ is a poset (=partially ordered set) with regards to the relation "$\succeq$" defined as follows: For any two elements (d_p, r_p) and (d_q, r_q). If $d_p \cap r_q \neq \emptyset$ then $(d_p, r_p) \succ (d_q, r_q)$, in words "$(d_p, r_p)$ is strictly greater than (d_q, r_q)". Moreover $(d_p, r_p) \succ (d_q, r_q)$ implies that $r_p \subset r_q$.*

5. *For any two elements (d_1, r_1) and (d_2, r_2) If $r_1 \cap r_2 \neq \emptyset$ and r_1, r_2 are not a subset of one another, then there is an element (d_3, r_3) in $K(D)$ such that $r_3 \subseteq r_1 \cap r_2$.*

As is easy to see the annotation, K in Fig. 2 does not satisfy the condition 4 of the lemma since the first element in K is bigger than the second but it's range is not a subset of the range of the second element. Therefore A is not DAG-representable.

REMARK 4. An algorithm is provided in [3, 2003a] that tests whether a given AG, possibly derived from a marginalized DAG relation, which satisfies the (necessary but not sufficient) conditions in Lemma 1 above, is DAG-representable. The main result of this work is to provide a polynomial algorithm which generates a "generalized annotated graph" representation (concept to be defined in the sequel) which is both compact and decidable. In some cases the generalized annotated graph reduces to a regular annotated graph which satisfies the condition of Lemma 1. If this is the case than, using the testing algorithm in [4, 2003b] we can check whether the given AG is DAG-representable. It is certainly not DAG-representable if the generalized annotated graph is not a regular AG or is a regular AG but does not satisfy the conditions of Lemma 1.

REMARK 5. When a given AG A is derived from a DAG then the annotation set $K = \{(d, r(d))\}$ can be interpreted as follows: The edge (a, b), in G, corresponding to d, (a moral edge) represents a conditional dependency. That is: there is some set of vertices, disjoint of $r(d)$, S_{ab} such that $(a_i b|S_{ab})$ is represented in A but a and b become dependent if any proper subset of $r(d)$ is observed i.e., $\neg(a; b|S)$ if $\emptyset \neq S \subseteq r(d)$.

In this paper we are concerned with the following problem: Given a relation which is represented by a generalized UG model and is not representable by a DAG (see [4, 2003]). Is it possible to find hidden variables such that the given relation results from the marginalization of the DAG representable relation over the expanded set of variables, including the hidden variables in addition to the given AG variables. We do not have a full solution to this problem which is so far an open problem. We present only a heuristic procedure, illustrated by several examples, for partially solving the problem.

3 PD-induced relations not representable as a marginalized DAG-representable relations-an example

While DAG's are widely used as a model that can represent PD-induced relations one may ask whether it might be possible to represent every PD-induced relation either by a DAG or, assuming the existence of latent variables, by a marginalized DAG. The answer to this question is negative as should be expected. A counterexample is given below.

Consider the following PD-induced relation, over 3 variable x, y, and z, consisting of two triplets only:

$$R = \{(x; y|\emptyset), (x; y|z) + \text{symmetric triplets}\}$$

Then R cannot be represented by a marginalized DAG. To prove this claim

assume that there is a DAG D with n variables, including x, y and z such that when D is marginalized over $\{x, y, z\}$, the marginalized DAG represents R. This assumption leads to a contradiction: Since $(x; z|\emptyset)$ and $(y; z|\emptyset)$ are not in R there must be trails π_{xz} and π_{yz} in D with no colliders included in them. Let π_{xy} be the concatenation of the two trails π_{xz} and π_{zy} (which is the trail π_{yz} reversed). Then π_{xy} connects between x and y and has no colliders on it except perhaps the vertex z. If z is a collider then $(x; y|z)$ is not represented in D. If z is not a collider then π_{xy} has no colliders on it and therefore $(x; y|\emptyset)$ is not represented in D. Therefore R cannot be represented by marginalizing D over $\{x, y, z\}$, a contradiction. That R is a PD-induced relation was shown by Milan Studeny [9, private communication, 2000] as follows:

Consider the PD over the binary variables x, y and the ternary variable z. The probability of the three variables for the different values of x, y, z is given below

$$\begin{aligned} p(0,0,0) &= p(0,0,1) = p(1,0,1) = p(1,0,2) = \tfrac{1}{8} \\ p(0,1,0) &= p(1,1,1) = \tfrac{1}{4} \\ p(x,y,z) &= 0 \text{ for all other configurations} \end{aligned}$$

The reader can convince himself that the relation induced by the above PD is the relation $R = \{(x; y|\emptyset), (x; y|z)\}$. Notice however that the relation R above is represented by the annotated graph below

$$G : x—z—y \qquad K = \{((x, y), \{z\})\}$$

see Paz [4, 2003b].

4 Finding Hidden Variables

In this section we consider the following problem. Given an AG which represents a relation that is not DAG representable, e.g. the relation represented by the AG shown in Fig. 2. Is it possible to find hidden variables which when added to the variables of the given relation will enable the representation of the given relation as a marginalized DAG representable relation, over the extra hidden variables. At this stage of the research we do not have an algorithmic procedure for solving this problem,a nd we do not have a characterization lemma, similar to Lemma 1 for AGs representing marginalized DAG representable relations. We can however present a heuristic procedure for tackling with this problem. We hope that it will be possible to extend the procedure into a full algorithm in the future. The procedure will be illustrated here by examples. The examples we will use however are such that we know in advance that the problem can be solved for them. This is due to the fact that we can not characterize so far the AG's for which the problem is solvable, as mentioned above. On the other hand we should keep in mind that not every PD-induced relation can be represented as a marginalized DAG-representable relation, as shown in the previous section.

4.1 Example 1

Consider the DAG D shown in Fig. 3. An equivalent AG, representing the same relation [4, 2003b] is shown in Fig. 4 where the broken lines (edges) represent conditional dependencies and correspond to uncoupled parents of colliders in Fig. 3.

Assume now that we want to marginalize the relation represented by the AG shown in Fig. 4 over the variables p and q. Using the procedure given in [6, 2006] we get a relation represented by the AG shown in Fig. 5 below.

The Derivation of the AG in Fig. 5 can be explained as follows:

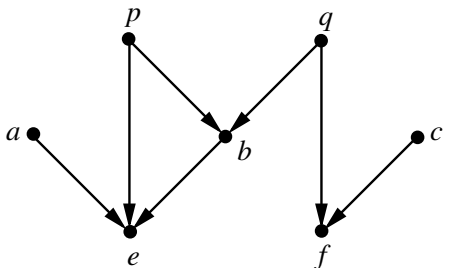


Figure 3. DAG D

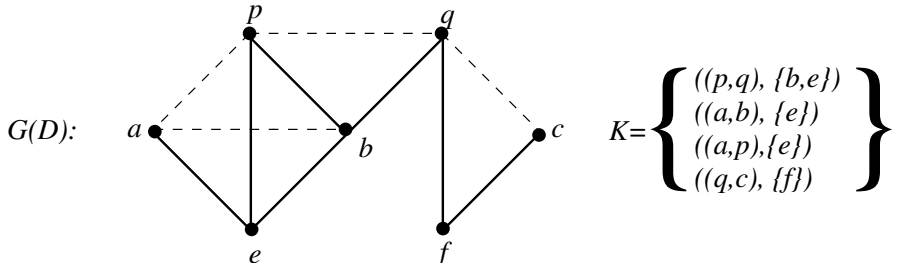


Figure 4. The annotated graph $A(D)$

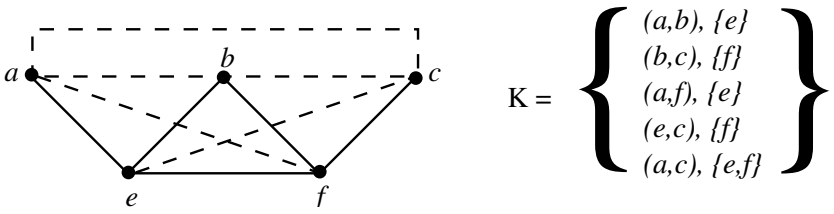


Figure 5. *AG* for the marginalized relation

- The solid edge b-f is induced by the path, in the AG shown in Fig. 4, b-q-f, through the marginalized variable q.
- Similarly, the solid edge e-f in Fig. 5 is induced by the path, in Fig. 4, $e - p - q - f$ through the marginalized variables p and q.
- The element $((a, b), \{e\})$ in Fig. 5 was transferred from Fig. 4 since it involves only non marginalized variables.
- The element $((b, c), \{f\})$ in Fig. 5 is induced by the path $b - q - c$ in Fig. 4 which is activated if f is observed.
- $((a, f)\{e\})$ is induced by the path $a - p - q - f$ which is activated in Fig. 4 if e is observed.
- $((e, c), \{f\})$ is induced by the path in Fig. 4 $e - p - q - c$ which is activated if f is observed.
- Finally $((a, c), \{e \wedge f\}$ is induced by the path $a - p - q - c$ in Fig. 4 which is activated if *both* e and f are observed.

REMARK 6. The dependency conditional shown in the fifth element of K in Fig. 5 is different from the conditionals in any AG representing DAG-representable relations in the following sense.The conditionals for DAG-representable relations consist of a set of variables such that, if any variable in the set is observed then the conditional is activated. In Fig. 5, in order to activate the conditional of the fifth element in K *both* variables e and f must be observed.

Assume now that we are given the AG shown in Fig. 5 with no prior knowledge that it represents a marginalized DAG-representable relation. We can see immediately that the relation represented in it is not DAG-representable. This follows from the remark above and also from the fact that K does not satisfy the necessary conditions of Lemma 1. E.g. the conditional $\{f\}$ in the second element is included in the pair $\{a, f\}$ of the third element but $\{e\}$ is not a subset of the conditional $\{f\}$ as required by Lemma 1, part 4.

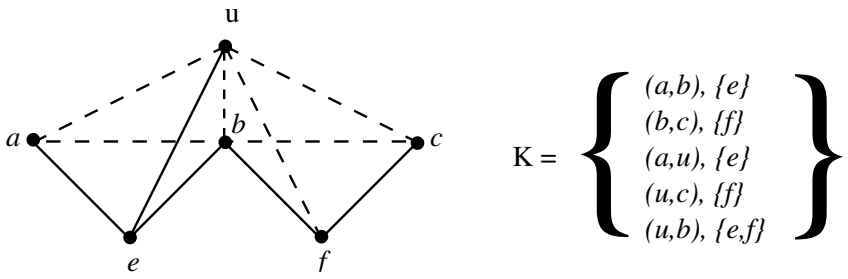


Figure 6. Extended AG

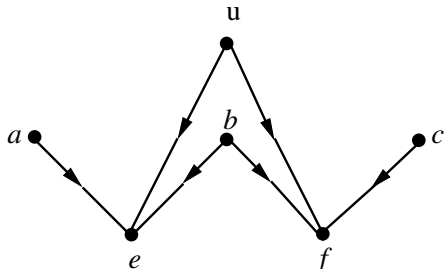


Figure 7. DAG which is equivalent to the AG in Fig. 6

Given the fact that the relation is not DAG-representable, we ask ourselves whether it is possible to add hidden variables to the variables of the relation such that the extended relation is DAG-representable and is such that when it is marginalized over the added (hidden) variables, it reduces to the given no DAG-representable relation. Consider first the fifth element (see Fig. 5) $(a, c)\{e \wedge f\}$ of K for the given relation. The conditional $\{e \wedge f\}$ of this element does not fit DAG-representable relation.

We notice that this element can be eliminated if we add a hidden variable$u$ that is connected by solid lines (unconditional dependant) to the variables e and f and is conditionally dependant on a with conditional e and is conditionally dependant on c with conditional f. The resulting graph is shown in Fig. 6.

The reader will easily convince himself that the annotation K shown in Fig. 6 fits the extended graph, where the first 2 elements are inherited from Fig. 5 and the other 3 elements are induced by the (hidden) new variable u.The solid edge (e, f) in Fig. 5 is removed in Fig. 6 since it is implied by the path $e - u - f$ when the extended relation is marginalized over u. The reader will also easily convince himself that if the relation shown in Fig. 6 is marginalized over u we get back the relation shown in Fig. 5. Moreover the annotation K in Fig. 6 complies with the necessary conditions of Lemma 1. Indeed the relation represented by the AG in Fig. 6 is DAG representable: Just direct all solid edges incident with e into e, direct all

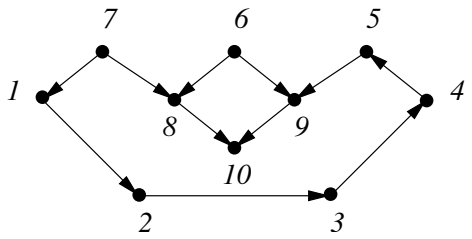


Figure 8. DAG for Example 2

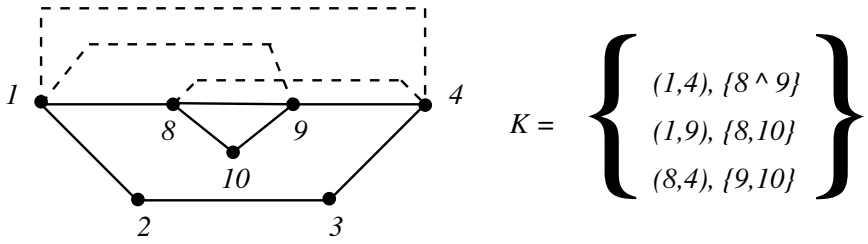


Figure 9. UG equivalent with the DAG in Fig. 8

solid edges incident with f into f and remove all broken edges. The result is shown in Fig. 7.

Notice that the DAG in Fig. 7 is quite different from the DAG shown in Fig. 3, but both reduce to the same relation when marginalized over their extra variables.

4.2 Example 2

Consider the DAG shown in Fig. 8. Using methods similar to the methods we used in the previous example we can get an equivalent AG which when marginalized over the vertices $5, 6$ and 7 results in the UG shown in Fig. 9.

Here again we can check and verify that K does not satisfy the conditions of Lemma 1, in particular the first element in K is a compound statement that does not fit DAG representable relations. So the relation represented by the AG in Fig. 9 is not DAG representable.Trying to eliminate the first element in K, we may assume the existence of a hidden variable u which is connected by solid edges to both 8 and 9, but is conditionally dependent on 1 with conditional 8 and is conditionally dependent on 4 with conditional 9. The resulting extended AG is shown in Fig. 10. The K in Fig. 10 satisfies the conditions of Lemma 1 and one can see that the AG in DAG is equivalent. Moreover marginalizing it over u reduces to the UG shown in Fig. 9. The equivalent DAG is shown in Fig. 11.

Notice that the DAG in Fig. 11 is quite different from the DAG in Fig. 8, but both result in the same AG when marginalizing over their corresponding extra variables.

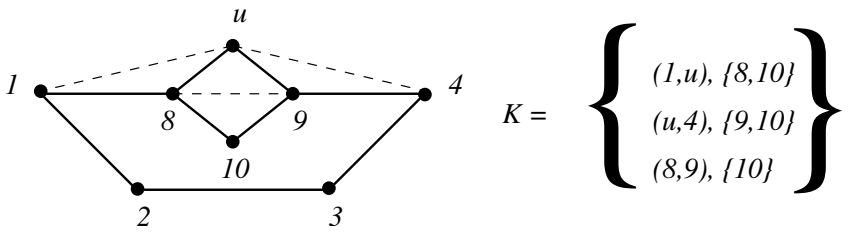


Figure 10. Expanded DAG equivalent AG

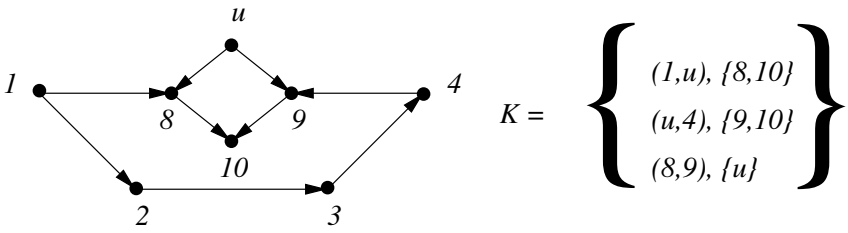


Figure 11. DAG equivalent to the AG in Fig. 10

4.3 Example 3

In this last example we consider the AG shown in Fig. 2 which is reproduced here, for convenience as Fig. 12.

While the K sets in the previous examples included elements with compound conditionals, both conditionals in this example are simple but the conditions of Lemma 1 are not satisfied: d in the first element is included in the second conditional and a of the second element is included in the first conditional, but the conditionals are no a subset of each other. Consider first the second element. We can introduce an additional variable u so that u will be conditionally dependent on b, and the element $(u, b), \{c, d\}$ will replace the element $(a, b), \{c, d\}$. The resulting AG is shown in Fig. 13 below.

It is easy to see that the graph in Fig. 13 reduces to graph in Fig. 12 when marginalized over u. We still need to take care of the first element since it is not satisfying the conditions of Lemma 1. We can now add additional new variable v and replace the first element $(b, d), \{a\}$ by the element $(u, v), \{a\}$. The resulting larger AG is shown in Fig. 14. Notice that the graph in Fig. 14 will reduce to the graph in Fig. 12.

To verify this we observe that marginalization over u and v induces the element $(b, d), \{a\}$ since when a and d are observed b gets connected to u (by d and the second element) and u is connected to v by the first element so that the path $b - u - v - d$ is activated through the extra variables (b, d and a exist in Fig. 12).

One can also verify that the AG in Fig. 14 is DAG equivalent after some simples

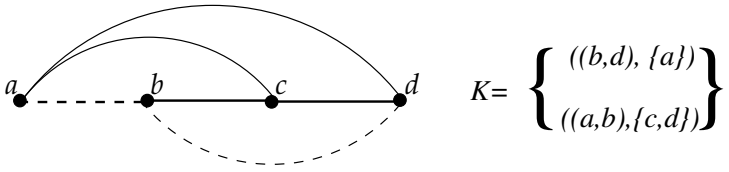


Figure 12. AG from Fig. 2

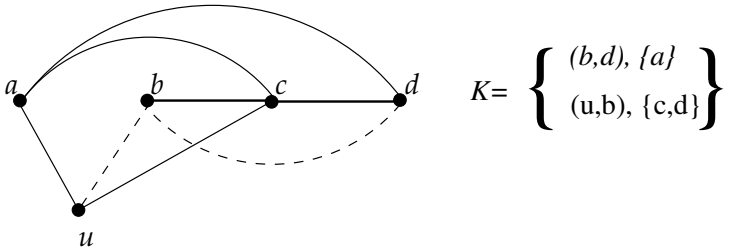


Figure 13. First extension of the AG in Fig. 12

modifications: we can remove the edges (a, c) and (a, d) since they are implied when marginalizing over u and v and we need to add the element $(v, c), \{d\}$ and a corresponding broken edge between v and c, which will be discarded when marginalizing. The equivalent DAG is shown in Fig. 15 and is identical with the DAG shown in Fig. 1.

Acknowledgment

I would like to thank the editors for allowing me to contribute this paper to the book honoring Judea Pearl, a friend and collaborator for more than 20 years now. Much of my scientific work in the past 20 years was influenced by discussions I had with him. His sharp intuition and his ability to pinpoint important problems and ask the right questions helped me in choosing the problems to investigate and finding their solutions, resulting in several scientific papers, including this one.

References

[1] S .L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

[2] S.L. Lauritzen. *Graphical Models*. Claredon, Oxford, U.K., 1996.

[3] A. Paz. An alternative version of Lauritzen et al's algorithm for checking representation of independencies. *Journal of Soft Computing*, pages 491–505, 2003a.

[4] A. Paz. The annotated graph model for representing DAG-representable relations – algorithmic approach. Technical Report Technical Report R-312,

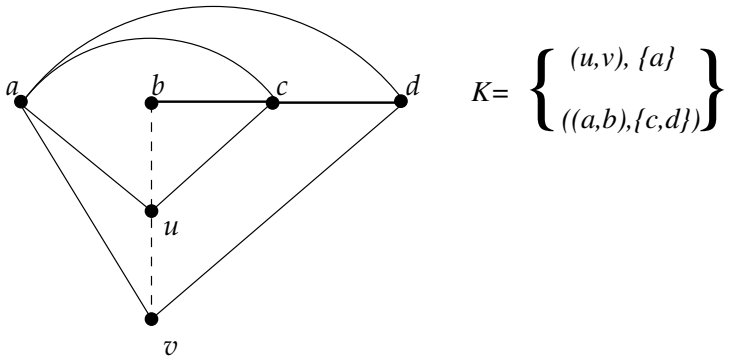


Figure 14. Extended AG

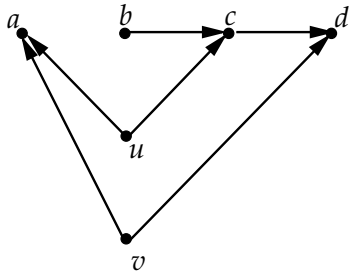


Figure 15. DAG which is equivalent to the AG in Fig. 14

Computer Science Department, UCLA, 2003b.

[5] A. Paz, R.Y. Geva, and M. Studeny. Representation of irrelevance relations by annotated graphs. *Fundamenta Informaticae*, 48:149–199, 2000.

[6] A. Paz. A New Graphical Model for the Representation of Marginalized DAG-Representable Relations. prodeedings of the 7th Workshop on Uncertainty Processing, pages 111–137, 2006, Mikulov, The Check Republic.

[7] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[8] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, 2000.

[9] M. Studeny, 2000. Private communication.

[10] T. Verma and J. Pearl. Equivalence and synthesis of causal models. In L.N. Kanal P. Bonissone, M. Henrion and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 225–268, B.V., 1991. Elsevier Science Publishers.

15

Probabilistic Programming Languages: Independent Choices and Deterministic Systems

David Poole

Pearl [2000, p. 26] attributes to Laplace [1814] the idea of a probabilistic model as a deterministic system with stochastic inputs. Pearl defines causal models in terms of deterministic systems with stochastic inputs. In this paper, I show how deterministic systems with (independent) probabilistic inputs can also be seen as the basis of modern probabilistic programming languages. Probabilistic programs can be seen as consisting of independent choices (over which there are probability distributions) and deterministic programs that give the consequences of these choices. The work on developing such languages has gone in parallel with the development of causal models, and many of the foundations are remarkably similar. Most of the work in probabilistic programming languages has been in the context of specific languages. This paper abstracts the work on probabilistic programming languages from specific languages and explains some design choices in the design of these languages.

Probabilistic programming languages have a rich history starting from the use of simulation languages such as Simula [Dahl and Nygaard 1966]. Simula was designed for discrete event simulations, and the built-in random number generator allowed for stochastic simulations. Modern probabilistic programming languages bring three extra features:

conditioning: the ability to make observations about some variables in the simulation and to compute the posterior probability of arbitrary propositions given these observations. The semantics can be seen in terms of rejection sampling: accept only the simulations that produce the observed values, but there are other (equivalent) semantics that have been developed.

inference: more efficient inference for determining posterior probabilities than rejection sampling.

learning: the ability to learn probabilities from data.

In this paper, I explain how we can get from Bayesian networks [Pearl 1988] to independent choices plus a deterministic system (by augmenting the set of variables). I explain the results from [Poole 1991; Poole 1993b], abstracted to be language independent, and show how they can form the foundations for a diverse set of probabilistic programming languages.

Consider how to represent a probabilistic model in terms of a deterministic system with independent inputs. In essence, given the probabilistic model, we construct a random variable for each free parameter of the original model. A deterministic system can be used to obtain the original variables from the new variables. There are two possible worlds structures, the original concise set of possible worlds in terms of the original variables, and the augmented set of possible worlds in terms of the new random variables. The dimensionality of the augmented space is the number of free parameters which is greater than the dimensionality of the original space (unless all variables were independent). However, the variables in the augmented worlds can be assumed to be independent, which makes them practical to use in a programming environment. The original worlds can be obtained using abduction.

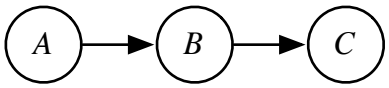
Independent choices with a deterministic programming language can be seen as the basis for most of the probabilistic programming languages, where the deterministic system can be a logic program [Poole 1993b; Sato and Kameya 1997; De Raedt, Kimmig, and Toivonen 2007], a functional program [Koller, McAllester, and Pfeffer 1997; Pfeffer 2001; Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum 2008], or even a low-level language like C [Thrun 2000].

There had been parallel developments in the development of causality [Pearl 2000], with causal models being deterministic systems with stochastic inputs. The augmented variables in the probabilistic programming languages are the variables needed for counterfactual reasoning.

1 Probabilistic Models and Deterministic Systems

In order to understand probabilistic programming languages, it is instructive to see how a probabilistic model in terms of a Bayesian network [Pearl 1988] can be represented as a deterministic system with probabilistic inputs.

Consider the following simple belief network, with Boolean random variables:



There are 5 free parameters to be assigned for this model; for concreteness assume the following values (where $A = true$ is written as a, and similarly for the other variables):

$$\begin{aligned} P(a) &= 0.1 \\ P(b|a) &= 0.8 \\ p(b|\neg a) &= 0.3 \\ P(c|b) &= 0.4 \\ p(c|\neg b) &= 0.75 \end{aligned}$$

To represent such a belief network in a probabilistic programming language, there are probabilistic inputs corresponding to the free parameters, and the programming

language specifies what follows from them. For example, in Simula [Dahl and Nygaard 1966], this could be represented as:

```
begin
   Boolean a,b,c;
   a := draw(0.1);
   if a then
      b := draw(0.8);
   else
      b := draw(0.3);
   if b then
      c := draw(0.4);
   else
      c := draw(0.75);
end
```

where $draw(p)$ is a Simula system predicate that returns true with probability p; each time it is called, there is an independent drawing.

Suppose c was observed, and we want the posterior probability of b. The conditional probability $P(b|c)$ is the proportion of those runs with c true that have b true. This could be computed using the Simula compiler by doing rejection sampling: running the program many times, and rejecting those runs that do not assign c to true. Out of the non-rejected runs, it returns the proportion that have b true. Of course, conditioning does not need to implemented that way; much of the development of probabilistic programming languages over the last twenty years is in devising more efficient ways to implement conditioning.

Another equivalent model to the Simula program can be given in terms of logic. There can be 5 random variables, corresponding to the independent draws, let's call them A, $Bifa$, $Bifna$, $Cifb$ $Cifnb$. These are independent with $P(a) = 0.1$, $P(bifa) = 0.8$, $P(bifna) = 0.3$, $P(cifb) = 0.4$, and $P(cifnb) = 0.75$. The other variables can be defined in terms of these:

$$b \iff (a \wedge bifa) \vee (\neg a \wedge bifna) \quad (1)$$

$$c \iff (b \wedge cifb) \vee (\neg b \wedge cifnbc) \quad (2)$$

These two formulations are essentially the same, they differ in how the deterministic system is specified, whether it is in Simula or in logic.

Any discrete belief network can be represented as a deterministic system with independent inputs. This was proven by Poole [1991, 1993b] and Druzdzel and Simon [1993]. These papers used different languages for the deterministic systems, but gave essentially the same construction.

2 Possible Worlds Semantics

A probabilistic programming language needs a specification of a deterministic system (given in some programming language) and a way to specify distributions over (independent) probabilistic inputs, or a syntactic variant of this. We will also assume that there are some observations, and that there are some query proposition for which we want the posterior probability.

In developing the semantics of a probabilistic programming language, we first define the set of possible worlds, and then a probability measure over sets of possible worlds [Halpern 2003].

In probabilistic programming, there are (at least) two sets of possible world that interact semantically. It is easiest to see these in terms of the above example. In the above belief network, there were three random variables A, B and C, which had complex inter-dependencies amongst them. With three binary random variables, there are 8 possible worlds. These eight possible worlds give a concise characterization of the probability distribution over these variables. I will call this the *concise* space of possible worlds.

In the corresponding probabilistic program, there is an augmented space with five inputs, each of which can be considered a random variable (these are A, $Bifa$, $Bifna$, $Cifb$ and $Cifnb$ in the logic representation). With five binary random variables, there are 32 possible worlds. The reason to increase the number of variables, and thus possible worlds, is that in this the *augmented* space, the random variables can be independent.

Note that the variables in the augmented space do not *have* to be independent. For example, $P(bifna|a)$ can be assigned arbitrarily since, when a is true, no other variable depends on $bifna$. In the augmented space, there is enough freedom to make the variables independent. Thus, we can arbitrarily set $P(bifna|a) = P(bifna|\neg a)$, which will be the same as $P(b|\neg a)$. The independence assumption makes the semantics and the computation simpler.

There are three semantics that could be given to a probabilistic program:

- The rejection sampling semantics; running the program with a random number generator, removing those runs that do not predict the observations, the posterior probability of a proposition is the limit, as the number of runs increases, of the proportion of the non-rejected runs that have the proposition true.

- The independent-choice semantics, where a possible world specifies the outcome of all possible draws. Each of these draws is considered to be independent. Given a world, the (deterministic) program would specify what follows. In this semantics, a possible world would select values for all five of the input variables in the example above, and thus gives rise to the augmented space of the above program with 32 worlds.

- The program-trace semantics, where a possible world specifies a sequence of outcomes for the draws encountered in one run of the program. In this semantics, a world would specify the values for three of the draws in the above program, as only three draws are encountered in any run of the program, and thus there would be 8 worlds.

In the logical definition of the belief network (or in the Simula definition if the draws are named), there are 32 worlds in the independent choice semantics:

World	A	$Bifa$	$Bifna$	$Cifb$	$Cifnb$	Probability
w_0	false	false	false	false	false	$0.9 \times 0.2 \times 0.7 \times 0.6 \times 0.25$
w_1	false	false	false	false	true	$0.9 \times 0.2 \times 0.7 \times 0.6 \times 0.75$
...						
w_{30}	true	true	true	true	false	$0.1 \times 0.8 \times 0.3 \times 0.4 \times 0.75$
w_{31}	true	true	true	true	true	$0.1 \times 0.8 \times 0.3 \times 0.4 \times 0.75$

The probability of each world is the product of the probability of each variable (as each of these variables is assumed to be independent). Note that in worlds w_{30} and w_{31}, the original variables A, B and C are all true; the value of $Cifnb$ is not used when B is true. These variables are also all true in the worlds that only differ in the value of $Bifna$, as again, $Bifna$ is not used when A is true.

In the program-trace semantics there are 8 worlds for this example, but not all of the augmented variables are defined in all worlds.

World	A	$Bifa$	$Bifna$	$Cifb$	$Cifnb$	Probability
w_0	false	$\bot$	false	$\bot$	false	$0.9 \times 0.7 \times 0.25$
w_1	false	$\bot$	false	$\bot$	true	$0.9 \times 0.7 \times 0.75$
...						
w_7	true	true	$\bot$	false	$\bot$	$0.1 \times 0.8 \times 0.6$
w_8	true	true	$\bot$	true	$\bot$	$0.1 \times 0.8 \times 0.4$

where $\bot$ means the variable is not defined in this world. These worlds cover all 8 cases of truth values for the original worlds that give values for A, B and C. The values of A, B and C can be obtained from the program. The idea is that a run of the program is never going to encounter an undefined value. The augmented worlds can be obtained from the worlds defined by the program trace by splitting the worlds on each value of the undefined variables. Thus each augmented world corresponds to a set of possible worlds, where the distinctions that are ignored do not make a difference in the probability of the original variables.

While it may seem that we have not made any progress, after all this is just a simple Bayesian network, we can do the same thing for any program with probabilistic inputs. We just need to define the independent inputs (often these are called *noise inputs*), and a deterministic program that gives the consequences of the choices of values for these inputs. It is reasonably easy to see that any belief network can be represented in this way, where the number of independent inputs is

equal to the number of free parameters of the belief network. However, we are not restricted to belief networks. The programs can be arbitrarily complex. We also do not need special "original variables", but can define the augmented worlds with respect to any variables of interest. Observations and queries (about which we want the posterior probability) can be propositions about the behavior of the program (e.g., that some assignment of the program variables becomes true).

When the language is Turing-equivalent, the worlds can be countably infinite, and thus there can be uncountably many worlds. A typical assumption is that the program eventually infers the observations and the query, that is, each run of the program will eventually (with probability 1) assign a truth value to any given observation and query. This is not always the case, such as when the query is to determine the fixed point of a Markov chain (see e.g., Pfeffer and Koller [2000]). We could also have non-discrete choices using continuous variables, which complicates but does not invalidate the discussion here.

A probability measure is over sets of possible worlds that form an algebra or a σ-algebra, depending on whether we want finite additivity or countable additivity [Halpern 2003]. For a programming language, we typically want countable additivity, as this allows us to not have a bound on the number of steps it takes to prove a query. For example, consider a person who plays the lottery until they win. The person will win eventually. This case is easy to represent as a probabilistic program, but requires reasoning about an infinite set of worlds.

The typical σ-algebra is the set of worlds that can be finitely described, and their (countable) union. Finitely describable means there is a finite set of draws that have their outcomes specified. Thus the probability measure is over sets of worlds that all have the same outcomes for a finite set of draws, and the union of such sets of worlds. We have a measure over such sets by treating the draws to be independent.

3 Abductive Characterization

Abduction is a form of reasoning characterized by "reasoning to the best explanation". It is typically characterized by finding a minimal consistent set of assumables that imply some observation. This set of assumables is called an *explanation* of the observation.

Poole [1991, 1993b] gave an abductive characterization of a probabilistic programming language, which gave a mapping between the independent possible world structure, and the descriptions of the worlds produced by abduction. This notion of abduction lets us construct a concise set of sets of possible worlds that is adequate to infer the posterior probability of a query.

The idea is that the the independent inputs become assumables. Given a probabilistic program, a particular observation obs and a query q, we characterize a (minimal) partition of possible worlds, where

- in each partition either $\neg obs$, $obs \wedge q$ or $obs \wedge \neg q$ can be inferred, and

- in each partition the same (finite) set of choices for the values of some of the inputs is made.

This is similar to the program-trace semantics, but will only need to make distinctions relevant to computing $P(q|obs)$. Given a probabilistic program, an observation and a query, the "explanations" of the observation conjoined with the query or its negation, produces such a partition of possible worlds.

In the example above, if the observation was $C = true$, and the query was B, we want the minimal set of assignments of values to the independent choices that gives $C = true \wedge B = true$ or $C = true \wedge B = false$. There are 4 such explanations:

- $A = true$, $Bifa = true$, $Cifb = true$
- $A = true$, $Bifa = false$, $Cifnb = true$
- $A = false$, $Bifna = true$, $Cifb = true$
- $A = false$, $Bifna = false$, $Cifnb = true$

The probability of each of these explanations is the product of the choices made, as these choices are independent. The posterior probability $P(B|C = true)$ can be easily computed by the weighted sum of the explanations in which B is true. Note also that the same explanations would be true even if C has unobserved descendants. As the number of descendants could be infinite if they were generated by a program, it is better to construct the finite relevant parts than prune the infinite irrelevant parts.

In an analogous way to how the probability of a real-variable is defined as a limit of discretizations, we can compute the posterior probability of a query given a probabilistic programming language. This may seem unremarkable until it is realized that for programs that are guaranteed to halt, there can be countably many possible worlds, and so there are uncountably many sets of possible worlds, over which to place a measure. For programs that are not guaranteed to halt, such as a sequence of lotteries, there are uncountably many possible worlds, and even more sets of possible worlds upon which to place a measure. Abduction gives us the sets of possible worlds in which to answer a conditional probability query. When the programs are not guaranteed to halt, the posterior probability of a query can be defined as the limit of the sets of possible worlds created by abduction, as long as the query can be derived in finite time for all but a set of worlds with measure zero.

In terms of the Simula program, explanations correspond to execution paths. In particular, an explanation corresponds to the outcomes of the draws in one trace of the program that infers the observations and a query or its negation. The set of traces of the program gives a set of possible worlds from which to compute probabilities.

When the program is a logic program, it isn't obvious what the program-trace semantics is. However, the semantics in terms of independent choices and abduction is well-defined. Thus it seems like the semantics in terms of abduction is more general than the program-trace semantics, as it is more generally applicable. It is also possible to define the abductive characterization independently of the details of the programming language, whereas defining a trace or run of a program depends on the details of the programming language.

Note that this abductive characterization is unrelated to MAP or MPE queries; we are defining the marginal posterior probability distribution over the query variables.

4 Inference

Earlier algorithms (e.g. Poole [1993a]) extract the minimal explanations and compute conditional probabilities from these. Later algorithms, such as used in IBAL [Pfeffer 2001], use sophisticated variable elimination to carry out inference in this space. IBAL's computation graph corresponds to a graphical representation of the explanations. Problog [De Raedt, Kimmig, and Toivonen 2007] compiles the computation graph into BDDs.

In algorithms that exploit the conditional independent structure, like variable elimination or recursive conditioning, the order that variables are summed out or split on makes a big difference to efficiency. In the independent choice semantics, there are more options available for summing out variables, thus there are more options available for making inference efficient. For example, consider the following fragment of a Simula program:

```
begin
   Boolean x;
   x := draw(0.1);
   if x then
      begin
        Boolean y := draw(0.2);
        ...
      end
   else
      begin
        Boolean z := draw(0.7);
        ...
      end
   ...
end
```

Here y is only defined when x is true and z is only defined when x is false. In the program-trace semantics, y and z are never both defined in any world. In

the independent-choice semantics, y and z are defined in all worlds. Efficiency considerations may mean that we want to sum out X first. In the independent-choice semantics, there is no problem, the joint probability on X and Y makes perfect sense. However, in the program trace semantics, it isn't clear what the joint probability of X and Y means. In order to allow for flexible elimination orderings in variable elimination or splitting ordering in recursive conditioning, the independent choice semantics is the natural choice.

Another possible way to implement probabilistic programming is to use MCMC [Milch, Marthi, Russell, Sontag, Ong, and Kolobov 2005; Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum 2008; McCallum, Schultz, and Singh 2009]. It is possible to do MCMC in either of the spaces of worlds above. The difference arises in conditionals. In the augmented space, for the example above, an MCMC state would include values for all of X, Y and Z. In the program-trace semantics, it would contain values for X and Y when $X = true$, and values for X and Z when $X = false$, as Y and Z are never simultaneously defined. Suppose X's value changes from $true$ to $false$. In the augmented space, it would just use the remembered values for Z. In the program-trace semantics, Z was not defined when Z was true, thus changing X from $true$ to $false$ means re-sampling all of the variables defined in that branch, including Z.

BLOG [Milch, Marthi, Russell, Sontag, Ong, and Kolobov 2005] and Church [Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum 2008] assign values to all of the variables in the augmented space. FACTORIE [McCallum, Schultz, and Singh 2009] works in what we have called the abductive space. Which is these is more efficient is an empirical question.

5 Learning Probabilities

The other aspect of modern probabilistic programming languages is the ability to learn the probabilities. As the input variables are rarely observed, the standard way to learn the probabilities is to use EM. Learning probabilities using EM in probabilistic programming languages is described by Sato [1995] and Koller and Pfeffer [1997]. In terms of available programming languages, EM forms the basis for learning in Prism [Sato and Kameya 1997; Sato and Kameya 2001], IBAL [Pfeffer 2001; Pfeffer 2007] and many subsequent languages.

One can do EM learning in either of the semantic structures. The difference is whether some data updates the probabilities of parameters that were not involved in computing the data. By making this choice explicit, it is easy to see that one should use the abductive characterization to only update the probabilities of the choices that were used to derive the data.

Structure learning for probabilistic programming languages has really only been explored in the context of logic programs, where the techniques of inductive logic programming can be applied. De Raedt, Frasconi, Kersting, and Muggleton [2008] overview this active research area.

6 Causal Models

It is interesting that the research on causal modelling and probabilistic programming languages have gone on in parallel, with similar foundations, but only recently have researchers started to combine them by adding causal constructs to probabilistic programming languages [Finzi and Lukasiewicz 2003; Baral and Hunsaker 2007; Vennekens, Denecker, and Bruynooghe 2009].

In some sense, the programming languages can be seen as representations for all of the counterfactual situations. A programming language gives a model when some condition is true, but also defines the "else" part of a condition; what happens when the condition is false.

In the future, I expect that programming languages will be the preferred way to specify causal models, and for interventions and counterfactual reasoning to become part of the repertoire of probabilistic programming languages.

7 Observation Languages

These languages can be used to compute conditional probabilities by having an "observer" (either humans or sensors) making observations of the world that are conditioned on. One problem that has long gone unrecognized is that it is often not obvious how to condition when the language allows for multiple individuals and relations among them. There are two main problems:

- The observer has to know what to specify and what vocabulary to use. Unfortunately, we can't expect an observer to "tell us everything that is observed". First, there are an unbounded number of true things one can say about a world. Second, the observer does not know what vocabulary to use to describe their observations of the world. As probabilistic models get more integrated into society, the models have to be able to use observations from multiple people and sensors. Often these observations are historic, or are created asynchronously by people who don't even know the model exists and are unknown when the model is being built.

- When there are are unique names, so that the observer knows which object(s) a model is referring to, an observer can provide a value to the random variable corresponding to the property of the individual. However, models often refer to roles [Poole 2007]. The problem is that the observer does not know which individual in the world fills a role referred to in the program (indeed there is often a probability distribution over which individuals fill a role). There needs to be some other mechanism other than asking for the observed value of a random variable or program variable, or the value of a property of an individual.

The first problem can be solved using ontologies. An ontology is a specification of the meaning of the symbols used in an information system. There are two main

areas that have spun off from the expert systems work of the 1970's and 1980's. One is the probabilistic revolution pioneered by Pearl. The other is often under the umbrella of knowledge representation and reasoning [Brachman and Levesque 2004]. A major aspect of this work is in the representation of ontologies that specify the meaning of symbols. An ontology language that has come to prominence recently is the language OWL [Hitzler, Krötzsch, Parsia, Patel-Schneider, and Rudolph 2009] which is one of the foundations of the semantic web [Berners-Lee, Hendler, and Lassila 2001]. There has recently been work on representing ontologies to integrate with probabilistic inference [da Costa, Laskey, and Laskey 2005; Lukasiewicz 2008; Poole, Smyth, and Sharma 2009]. This is important for Bayesian reasoning, where we need to condition on all available evidence; potentially applicable evidence is (or should be) published all over the world. Finding and using this evidence is a major challenge. This problem in being investigated under the umbrella of semantic science [Poole, Smyth, and Sharma 2008].

To understand the second problem, suppose we want to build a probabilistic program to model what apartments people will like for an online apartment finding service. This is an example where models of what people want and descriptions of the world are built asynchronously. Rather than modelling people's preferences, suppose we want to model whether they would want to move in and be happy there in 6 months time (this is what the landlord cares about, and presumably what the tenant wants too). Suppose Mary is looking for an apartment for her and her daughter, Sam. Whether Mary likes an apartment depends on the existence and the properties of Mary's bedroom and of Sam's bedroom (and whether they are the same room). Whether Mary likes a room depends on whether it is large. Whether Sam likes a room depends on whether it is green. Figure 1 gives one possible probability model, using a belief network, that follows the above story.

If we observe a particular apartment, such as the one on the right of Figure 1, it isn't obvious how to condition on the observations to determine the posterior probability that the apartment is suitable for Mary. The problem is that apartments don't come labelled with Mary's bedroom and Sam's bedroom. We need some role assignment that specifies which bedroom is Mary's and which bedroom is Sam's. However, which room Sam chooses depends on the colour of the room. We may also like to know the probability that a bachelor's apartment (that contains no bedrooms) would be suitable.

To solve the second problem, we need a representation of observations. These observations and the programs need to refer to interoperating ontologies. The observations need to refer to the existence of objects, and so would seem to need some subset of the first-order predicate calculus. However, we probably don't want to allow arbitrary first-order predicate calculus descriptions of observations. Arguably, people do not observe arbitrary disjunctions. One simple, yet powerful, observation language, based on RDF [Manola and Miller 2004] was proposed by Sharma, Poole, and Smyth [2010]. It is designed to allow for the specification of observations of

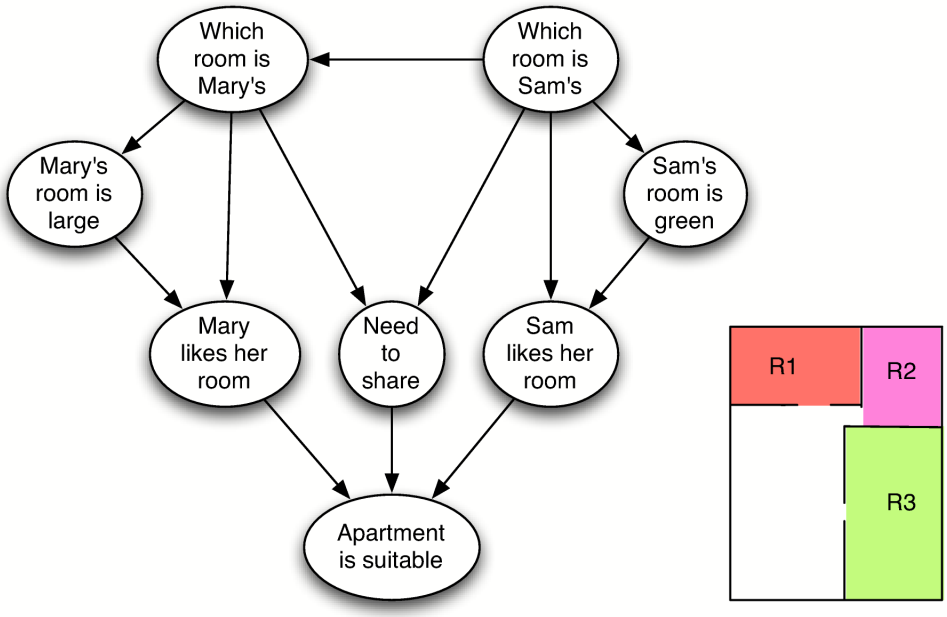


Figure 1. A possible belief network and observed apartment for the apartment example

the existence and non-existence of objects, and the properties and non-properties of objects. Observations are represented in terms of quadruples of the form:

$$\langle object, property, value, truthvalue \rangle$$

When the range of *property* is a fixed set, this quadruple means $property(object, value)$ or $\neg property(object, value)$, depending on the truth value.

When *value* is a world object, this quadruple means $\exists value\ property(object, value)$ or $\neg\exists value\ property(object, value)$, where the other mentions of *value* are in the scope of the quantification. This simple language seems to have the power to represent real observations, without representing arbitrary first-order formulae. It is then part of the program or the programming language to determine the correspondence between the objects in the language and the observed objects.

In the above example, the apartment can be described in terms of the existence of three bedrooms, one medium-sized and red (which we call $R1$), one small and pink (which we call $R2$) one large and green (which we will call $R3$). We also observe

that there is not a fourth bedroom. This can be represented as:

$$\langle apr, hasBedroom, R1, true\rangle$$
$$\langle R1, size, medium, true\rangle$$
$$\langle R1, color, red, true\rangle$$
$$\langle apr, hasBedroom, R2, true\rangle$$
$$\ldots$$
$$\langle apr, hasBedroom, R4, false\rangle$$

Thus this language is analogous to observing conjunctions of propositional atoms. However, it also lets us observe the existence and non-existence of objects, without allowing for representing arbitrary disjunctions.

Such observational languages are an important complement to probabilistic programming languages.

8 Pivotal Probabilistic Programming Language References

Probabilistic Horn abduction [Poole 1991; Poole 1993b] is the first language with a probabilistic semantics that allows for conditioning. Much of the results of this paper were presented there, in the context of logic programs. Probabilistic Horn abduction was refined into the Independent Choice Logic [Poole 1997] that allowed for choices made by multiple agents, and there is a clean integration with negation as failure [Poole 2000]. Prism introduced learning into essentially the same framework [Sato and Kameya 1997; Sato and Kameya 2001]. More recently, Problog [De Raedt, Kimmig, and Toivonen 2007] has become a focus to implement many logical languages into a common framework.

In parallel to the work on probabilistic logic programming languages, has been work on developing probabilistic functional programming languages starting with Stochastic Lisp [Koller, McAllester, and Pfeffer 1997], including IBAL [Pfeffer 2001; Pfeffer 2007], A-Lisp [Andre and Russell 2002] and Church [Goodman, Mansinghka, Roy, Bonawitz, and Tenenbaum 2008].

Other probabilistic programming languages are based on more imperative languages such as CES [Thrun 2000], based on C, and the languages BLOG [Milch, Marthi, Russell, Sontag, Ong, and Kolobov 2005] and FACTORIE [McCallum, Schultz, and Singh 2009] based on object-oriented languages. BLOG concentrates on number and identity uncertainty, where the probabilistic inputs include the number of objects and whether two names refer to the same object or not.

9 Conclusion

This paper has concentrated on similarities, rather than the differences, between the probabilistic programing languages. Much of the research in the area has concentrated on specific languages, and this paper is an attempt to put a unifying structure on this work, in terms of independent choices and abduction.

Unfortunately, it is difficult to implement an efficient learning probabilistic programming language. Most of the languages that exist have just one implementation; the one developed by the designers of the language. As these are typically research code, the various implementations have concentrated on different aspects. For example, the Prism implementation has concentrated on incorporating different learning algorithms, the IBAL implementation has concentrated on efficient inference, my AILog2 implementation of ICL has concentrated on debugging and explanation and use by beginning students[1]. Fortunately, many of the implementations are publicly available and open-source, so that they are available for others to modify.

One of the problems with the current research is that the language and the implementation are often conflated. This means that researchers feel the need to invent a new language in order to investigate a new learning or inference technique. For example, the current IBAL implementation uses exact inference, but it does not need to; different inference procedures could be used with the same language. If we want people to use such languages, they should be able to take advantage of the advances in inference or learning techniques without changing their code. One interesting project is the ProbLog project [De Raedt, Kimmig, and Toivonen 2007], which is building an infrastructure so that many of the different logic programming systems can be combined, and so that the user can use a standard language, and it can incorporate advances in inference and learning.

Probabilistic programming languages have an exciting future. We will want to have rich languages to specify causal mechanisms, processes, and rich models. How to program these models, learn them, and efficiently implement these are challenging research problems.

Acknowledgments: Thanks to Peter Carbonetto, Mark Crowley, Jacek Kisyński and Daphne Koller for comments on earlier versions of this paper. Thanks to Judea Pearl for bringing probabilistic reasoning to the forefront of AI research. This work could not have been done without the foundations he lay. This work was supported by an NSERC discovery grant to the author.

References

Andre, D. and S. Russell (2002). State abstraction for programmable reinforcement learning agents. In *Proc. AAAI-02*.

Baral, C. and M. Hunsaker (2007). Using the probabilistic logic programming language P-log for causal and counterfactual reasoning and non-naive conditioning. In *Proc. IJCAI 2007*, pp. 243–249.

Berners-Lee, T., J. Hendler, and O. Lassila (2001). The semantic web: A new

[1] We actually use it to teach logic programming to beginning students. They use it for assignments before they learn that it can also handle probabilities. The language shields the students from the non-declarative aspects of languages such as Prolog, and has many fewer built-in predicates to encourage students to think about the problem they are trying to solve.

form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American May*, 28–37.

Brachman, R. and H. Levesque (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann.

da Costa, P. C. G., K. B. Laskey, and K. J. Laskey (2005, Nov). PR-OWL: A Bayesian ontology language for the semantic web. In *Proceedings of the ISWC Workshop on Uncertainty Reasoning for the Semantic Web*, Galway, Ireland.

Dahl, O.-J. and K. Nygaard (1966). Simula : an ALGOL-based simulation language. *Communications of the ACM 9*(9), 671–678.

De Raedt, L., P. Frasconi, K. Kersting, and S. H. Muggleton (Eds.) (2008). *Probabilistic Inductive Logic Programming*. Springer.

De Raedt, L., A. Kimmig, and H. Toivonen (2007). ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pp. 2462–2467.

Druzdzel, M. and H. Simon (1993). Causality in Bayesian belief networks. In *Proc. Ninth Conf. on Uncertainty in Artificial Intelligence (UAI-93)*, Washington, DC, pp. 3–11.

Finzi, A. and T. Lukasiewicz (2003). Structure-based causes and explanations in the independent choice logic. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003)*, Acapulco, Mexico, pp. 225–232.

Goodman, N., V. Mansinghka, D. M. Roy, K. Bonawitz, and J. Tenenbaum (2008). Church: a language for generative models. In *Proc. Uncertainty in Artificial Intelligence (UAI)*.

Halpern, J. Y. (2003). *Reasoning about Uncertainty*. Cambridge, MA: MIT Press.

Hitzler, P., M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph (2009). *OWL 2 Web Ontology Language Primer*. W3C.

Koller, D., D. McAllester, and A. Pfeffer (1997). Effective Bayesian inference for stochastic programs. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, Providence, Rhode Island, pp. 740–747.

Koller, D. and A. Pfeffer (1997). Learning probabilities for noisy first-order rules,. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, Japan, pp. 1316–1321.

Laplace, P. S. (1814). *Essai philosophique sur les probabilities*. Paris: Courcier. Reprinted (1812) in English, F.W. Truscott amd F. L. Emory (Trans.) by Wiley, New York.

Lukasiewicz, T. (2008). Expressive probabilistic description logics. *Artificial Intelligence 172*(6-7), 852–883.

Manola, F. and E. Miller (2004). *RDF Primer*. W3C Recommendation 10 February 2004.

McCallum, A., K. Schultz, and S. Singh (2009). Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems Conference (NIPS)*.

Milch, B., B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov (2005). BLOG: Probabilistic models with unknown objects. In *Proc. 19th International Joint Conf. Artificial Intelligence (IJCAI-05)*, Edinburgh.

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

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Pfeffer, A. (2001). IBAL: A probabilistic rational programming language. In *Proc. 17th International Joint Conf. Artificial Intelligence (IJCAI-01)*.

Pfeffer, A. (2007). The design and implementation of IBAL: A general-purpose probabilistic language. In L. Getoor and B. Taskar (Eds.), *Statistical Relational Learning*. MIT Press.

Pfeffer, A. and D. Koller (2000). Semantics and inference for recursive probability models,. In *National Conference on Artificial Intelligence (AAAI)*.

Poole, D. (1991, July). Representing Bayesian networks within probabilistic Horn abduction. In *Proc. Seventh Conf. on Uncertainty in Artificial Intelligence (UAI-91)*, Los Angeles, pp. 271–278.

Poole, D. (1993a). Logic programming, abduction and probability: A top-down anytime algorithm for computing prior and posterior probabilities. *New Generation Computing 11*(3–4), 377–400.

Poole, D. (1993b). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence 64*(1), 81–129.

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence 94*, 7–56. special issue on economic principles of multi-agent systems.

Poole, D. (2000). Abducing through negation as failure: stable models in the Independent Choice Logic. *Journal of Logic Programming 44*(1–3), 5–35.

Poole, D. (2007, July). Logical generative models for probabilistic reasoning about existence, roles and identity. In *22nd AAAI Conference on AI (AAAI-07)*.

Poole, D., C. Smyth, and R. Sharma (2008). Semantic science: Ontologies, data and probabilistic theories. In P. C. da Costa, C. d'Amato, N. Fanizzi, K. B.

Laskey, K. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool (Eds.), *Uncertainty Reasoning for the Semantic Web I*, LNAI/LNCS. Springer.

Poole, D., C. Smyth, and R. Sharma (2009, Jan/Feb). Ontology design for scientific theories that make probabilistic predictions. *IEEE Intelligent Systems 24*(1), 27–36.

Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP95)*, Tokyo, pp. 715–729.

Sato, T. and Y. Kameya (1997). PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI97)*, pp. 1330–1335.

Sato, T. and Y. Kameya (2001). Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research (JAIR) 15*, 391–454.

Sharma, R., D. Poole, and C. Smyth (2010). A framework for ontologically-grounded probabilistic matching. *International Journal of Approximate Reasoning In press*.

Thrun, S. (2000). Towards programming tools for robots that integrate probabilistic computation and learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA. IEEE.

Vennekens, J., M. Denecker, and M. Bruynooghe (2009). CP-logic: A language of causal probabilistic events and its relation to logic programming. *Theory and Practice of Logic Programming (TPLP) to appear*.

16

Arguing with a Bayesian Intelligence

Ingrid Zukerman

1 Introduction

Bayesian Networks (BNs) [Pearl 1988] constitute one of the most influential advances in Artificial Intelligence, with applications in a wide range of domains, e.g., meteorology, agriculture, medicine and environment. To further capitalize on its clear technical advantages, a Bayesian intelligence (a computer system that employs a BN as its knowledge representation and reasoning formalism) should be able to communicate with its users, i.e., users should be able to put forward their views, and the system should be able to generate responses in turn. However, communication between a Bayesian and a human intelligence poses some challenges, as people generally do not engage in normative probabilistic reasoning when faced with uncertainty [Evans, Barston, and Pollard 1983; Lichtenstein, Fischhoff, and Phillips 1982; Tversky and Kahneman 1982]. In addition, human discourse is typically *enthymematic* (i.e., it omits easily inferable information), and usually the beliefs and inference patterns of conversational partners are not perfectly synchronized. As a result, an addressee's understanding may differ from the message intended by his or her conversational partner.

In this chapter, we offer a mechanism that enables a Bayesian intelligence to interpret human arguments for or against a proposition. This mechanism, which is implemented in a system called BIAS (*Bayesian Interactive Argumentation System*), constitutes a building block of a future system that will enable a Bayesian reasoner to communicate with people.[1]

In order to address the above challenges, we adopt the view that discourse interpretation is the process of integrating the contribution of a conversational partner into the addressee's mental model [Kashihara, Hirashima, and Toyoda 1995; Kintsch 1994], which in BIAS's case is a BN. Notice, however, that when performing such an integration, one cannot be sure that one is drawing the intended inferences or reinstating the exact information omitted by the user. All an addressee can do is construct an account of the conversational partner's discourse that makes sense to him or her. An interpretation of an argument that makes sense to BIAS is a subnet of its BN and a set of beliefs.

To illustrate these ideas, consider the argument in Figure 1(a) regarding the guilt

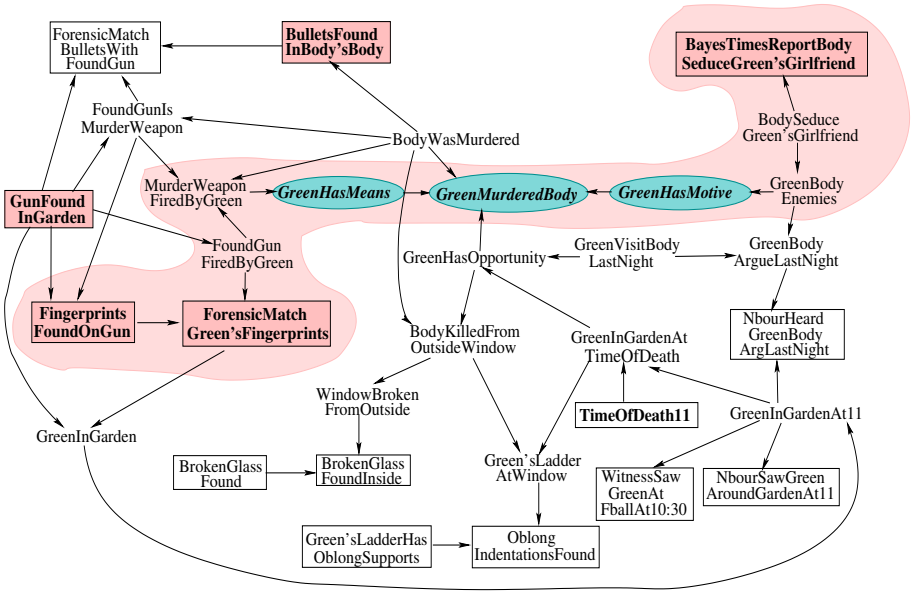
[1] The complementary building block, a mechanism that generates arguments from BNs, is described in [Korb, McConachy, and Zukerman 1997; Zukerman, McConachy, and Korb 1998].

***Fingerprints being found on the gun**, and **forensics matching the fingerprints with Mr Green** implies that Mr Green probably had the means to murder Mr Body.*

***The Bayesian Times reporting that Mr Body seduced Mr Green's girlfriend** implies that Mr Green possibly had a motive to murder Mr Body.*

Since Mr Green probably had the means to murder Mr Body, and Mr Green possibly had a motive to murder Mr Body, then Mr Green possibly murdered Mr Body.

(a) Sample argument



(b) BN and argument interpretation

Figure 1. Argument, domain BN and interpretation of the argument

of Mr Green in the murder of Mr Body (this argument is a gloss of an argument entered through a web interface).[2] The argument is interpreted in the context of the BN in Figure 1(b), which depicts the scenario for this murder mystery — the domain where we tested our ideas [Zukerman and George 2005]. The light-shaded bubble demarcates the BN subnet corresponding to the interpretation preferred by BIAS.[3] This interpretation contains propositions in the BN which bridge the

[2]We use the following linguistic terms, which are similar to those used in [Elsaesser 1987], to convey degree of belief: *Very Probable, Probable, Possible* and their negations, and *Even Chance*. According to our surveys, these terms are the most consistently understood by people [George, Zukerman, and Niemann 2007].

[3]The observable evidence nodes are boxed, and the evidence nodes that were actually observed by the user are boldfaced, as are the evidence nodes employed in the argument. The

reasoning gaps in the user's (enthymematic) argument.

Our approach casts the problem of finding a good interpretation of a user's argument as a model selection problem, where the interpretation is the model and the argument is the data. The criterion for selecting an interpretation is inspired by the Minimum Message Length (MML) principle [Wallace 2005] — an operational form of Occam's Razor that balances *model complexity* against *data fit*. That is, we aim to select *the simplest model (interpretation) that explains well the observed data (argument)*. The complexity of a model may be viewed as its probability in light of background knowledge: models that depart from the background knowledge are less probable than models that match the background knowledge, and structurally complex models are less probable than simpler models. Data fit may be viewed as the probability of the data given the model, i.e., the probability that a user who intended a particular interpretation presented the given argument.

The model selection problem is represented as a search problem, where a search procedure generates alternative interpretations, and an evaluation function assesses the merit of each interpretation. Since interpretations must be generated in real time, we use an (almost) anytime algorithm [Dean and Boddy 1988; Horvitz, Suermondt, and Cooper 1989] as our search procedure (Section 3). Our evaluation function is a probabilistic formulation of key aspects of the MML principle (Section 4).

This chapter is organized as follows. In the next section, we define an interpretation. Our algorithm for postulating interpretations is described in Section 3, and our probabilistic formulation for assessing an interpretation in Section 4. In Section 5 we present results of our evaluations, followed by a discussion of the limitations of our system, and advances required to support practical Bayesian argumentation systems.

2 What is an Interpretation?

As mentioned in Section 1, we view the interpretation of an argument as a "self explanation" — an account of the argument that makes sense to the addressee. For BIAS, such an account is specified by a tuple $\{IG, SC, EE\}$, where IG is an *interpretation graph*, SC is a *supposition configuration*, and EE are *explanatory extensions*.[4]

- An **interpretation graph** is a subnet of the domain BN that connects the propositions in an argument. This subnet bridges inferential leaps in the argument, but the bridges so constructed may not be those intended by the user.

nodes corresponding to the consequents in the user's argument (GreenHasMeans, GreenHasMotive and GreenMurderedBody) are italicized and oval-shaded.

[4]In our initial work, our interpretations contained only interpretation graphs [Zukerman and George 2005]. Subsequent trials with users demonstrated the need for supposition configurations and explanatory extensions [George, Zukerman, and Niemann 2007].

- A **supposition configuration** is a set of beliefs attributed to the user to account for the beliefs expressed in the argument. A supposition may maintain a belief shared with the system (i.e., nothing is supposed), instantiate a node in a BN to TRUE or FALSE, or uninstantiate (forget) a previously instantiated node.

- **Explanatory extensions** consist of nodes and links that are added to an interpretation graph in order to make the inferences in the interpretation more acceptable to people (in early trials of the system, inferences were deemed unacceptable if they contained increases in certainty or large jumps in belief between the antecedents and the consequent). Contrary to suppositions, the beliefs in explanatory extensions are shared by the user and the system.

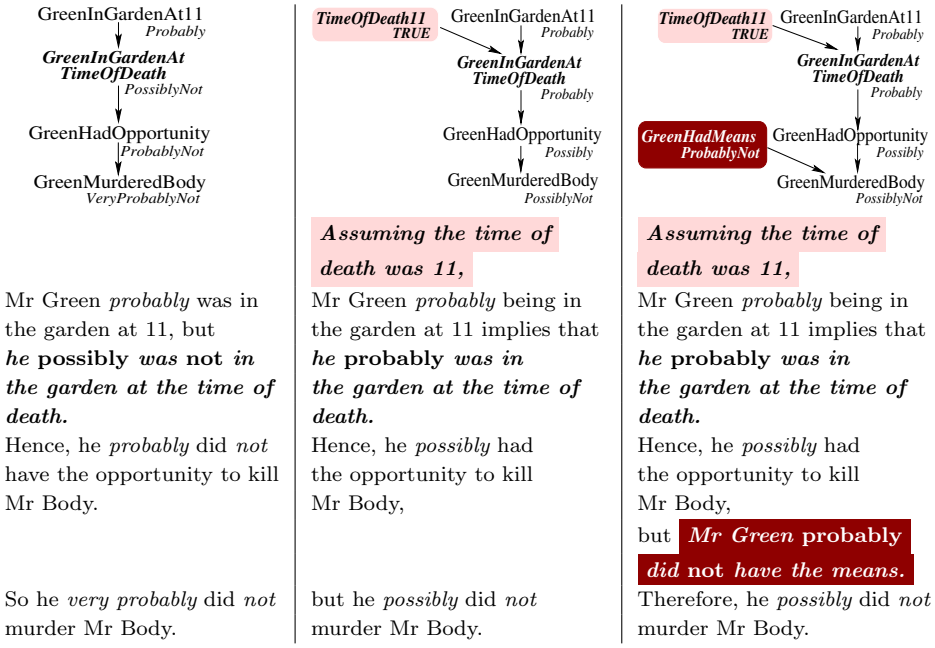
To illustrate these components, consider the brief argument in Figure 2(a) in relation to our murder mystery, and the three segments under it. Each segment, which highlights one of these components, shows the Bayesian subnet corresponding to the preferred interpretation and its textual rendition. Figure 2(b) shows the interpretation graph alone (the node that connects between the propositions in the argument appears in boldface italics); Figure 2(c) adds a supposition to the interpretation (in a shaded box); and Figure 2(d) adds an explanatory extension (white text in a dark box).

Let us now examine in more detail the three segments in Figure 2. Only one proposition (GreenInGardenAtTimeOfDeath, in boldface italics) needs to be added to connect the argument propositions in the domain BN, and create the interpretation graph in Figure 2(b). Note that the beliefs in this interpretation graph (obtained by Bayesian propagation of the system's beliefs in the domain BN) do not match those in the argument. The argument antecedent GreenInGardenAt11 yields a belief of *PossiblyNot* in GreenInGardenAtTimeOfDeath, which in turn implies that Mr Green *ProbablyNot* had the opportunity to kill Mr Body, and *VeryProbablyNot* committed the murder. To address this problem, the system *supposes* that the user believes that TimeOfDeath11 is TRUE (instead of the system's belief of *Probably*). Figure 2(c) shows how this supposition (in a shaded box) fits in the interpretation graph, and depicts its impact on the beliefs in the interpretation. These beliefs now match those in the argument. However, now the last inference in the argument goes from Mr Green *Possibly* having the opportunity to kill Mr Body to Mr Green *PossiblyNot* murdering Mr Body — a "jump in belief" which people find unacceptable. This problem is addressed by an *explanatory extension* that justifies the consequent on the basis of beliefs presumably shared with the user.[5] In this case, the selected proposition is that Mr Green *ProbablyNot* had the means to murder Mr Body. Figure 2(d) shows how this explanatory extension (white text in a dark box) fits in

[5]Note that the interpretation graph in Figure 2(a) also requires explanatory extensions for all the inferences (to overcome the jump in belief in the first inference, and the increases in certainty in the next two inferences). We omitted these explanatory extensions for clarity of exposition.

Mr Green probably being in the garden at 11 implies that he possibly had the opportunity to kill Mr Body, but he possibly did not murder Mr Body.

(a) Sample argument



(b) Interpretation graph (c) Interpretation graph + Supposition config. (d) Interpretation graph + Supposition config. + Explanatory extension

Figure 2. Interpretation graph, supposition configuration and explanatory extension

the interpretation graph. Note that explanatory extensions do not affect the beliefs in an interpretation, as they simply state previously held beliefs.

3 Proposing Interpretations

The problem of finding the best interpretation is exponential, as there are many candidates for each component of an interpretation, and complex interactions between supposition configurations and interpretation graphs. For example, making a supposition could invalidate an otherwise sound line of reasoning.

In order to generate reasonable interpretations in real time, we apply Algorithm 1 — an (almost) anytime algorithm [Dean and Boddy 1988; Horvitz, Suermondt, and Cooper 1989] that iteratively proposes interpretations until time runs out, i.e., until the system has to act upon a preferred interpretation or show the user one or more interpretations for validation [George, Zukerman, and Niemann 2007; Zukerman and George 2005]. At present, our interaction with the user stops when interpretations

Algorithm 1 Argument Interpretation

Require: User argument, domain knowledge BN

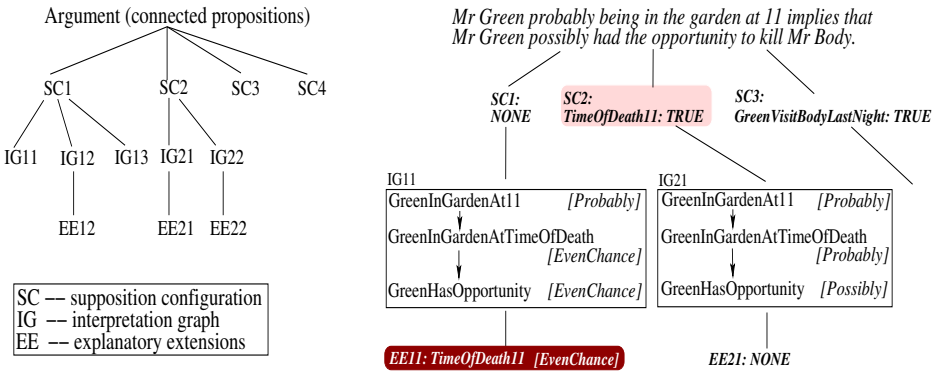
1: **while** there is time **do**
2: Propose a supposition configuration SC_i — this can be null, an existing supposition configuration or a new one.
3: Propose a new interpretation graph IG_{ij} under supposition configuration SC_i.
4: Propose explanatory extensions EE_{ij} for interpretation graph IG_{ij} under supposition configuration SC_i as necessary.
5: Estimate the probability of interpretation $\{SC_i, IG_{ij}, EE_{ij}\}$.
6: Retain the top K most probable interpretations.
7: **end while**
8: Present the retained interpretations to the user for validation.

are presented for validation. However, in a complete system, a dialogue module would have to determine a course of action based on the generated interpretations.

In each iteration, the algorithm proposes an interpretation which consists of a supposition configuration, an interpretation graph and explanatory extensions (Steps 2-4). It then estimates the probability of this interpretation (Step 5), and retains the top K most probable interpretations (Step 6). The procedure for building interpretation graphs is described in [Zukerman and George 2005], and the procedures for postulating supposition configurations and generating explanatory extensions are described in [George, Zukerman, and Niemann 2007]. Here we outline the general interpretation process and briefly describe these procedures.

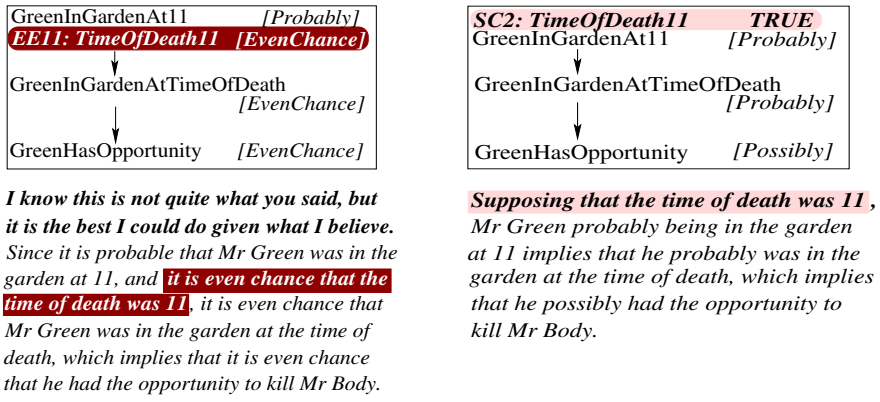
Figure 3(a) depicts a portion of the search tree generated by our algorithm, with multiple supposition configurations considered in the first level, multiple interpretation graphs in the second level, and one set of explanatory extensions per interpretation graph in the third level. A supposition configuration is proposed first because suppositions change the beliefs in the domain and affect the manner in which beliefs influence each other. This happens because suppositions are implemented as instantiations or uninstantiations of nodes, which may block a path in a BN (precluding the propagation of evidence through this path), or unblock a previously blocked path. These interactions, which are difficult to predict until an interpretation graph is complete, motivate the large number of alternatives considered in the first two levels of the search tree. In contrast, explanatory extensions do not seem to have complex interactions with interpretation graphs. Hence, they are generated deterministically in the third level of the search tree — only one set of explanatory extensions is proposed for each interpretation.

Figure 3(b) shows a portion of the search tree instantiated for the short argument at the root node of this tree: "*Mr Green probably being in the garden at 11 implies that Mr Green possibly had the opportunity to kill Mr Body*". In this example, the



(a) Search tree in progress

(b) Sample search tree



(c) Interpretation $(SC1, IG11, EE11)$

(d) Interpretation $(SC2, IG21, EE21)$

Figure 3. Argument interpretation process

user's belief in the consequent of the argument differs from the belief obtained by BIAS by means of Bayesian propagation from the evidence nodes in the domain BN. As indicated above, BIAS attempts to address this problem by making suppositions about the user's beliefs. The first level of the sample search tree in Figure 3(b) contains three supposition configurations $SC1$, $SC2$ and $SC3$. $SC1$ posits no beliefs that differ from those in the domain BN, thereby retaining the mismatch between the user's belief in the consequent and BIAS's belief; $SC2$ posits that the user believes that the time of death is 11; and $SC3$ posits that the user believes that Mr Green visited Mr Body last night.

The best interpretation graph for $SC1$ is $IG11$ (the evaluation of the goodness of an interpretation is described in Section 4). Here the belief in the consequent differs from that stated by the user, prompting the generation of a preface that acknowledges this fact. In addition, the interpretation graph has a large jump in belief (from *Probably* to *EvenChance*), which causes BIAS to add the mutually believed proposition TimeOfDeath11*[EvenChance]* as an explanatory extension. The

resultant interpretation and its gloss appear in Figure 3(c). The best interpretation graph for $SC2$ is $IG21$, which matches the beliefs in the user's argument. The resultant interpretation and its gloss appear in Figure 3(d). Note that both $(SC1, IG11, EE11)$ and $(SC2, IG21, EE21)$ mention TimeOfDeath11, but in the first interpretation this proposition is used as an explanatory extension (with a belief of *EvenChance* obtained by Bayesian propagation), while in the second interpretation it is used as a supposition (with a belief of TRUE). Upon completion of this process, BIAS retains the K most probable interpretations. In this example, the best interpretation is $\{SC2, IG21, EE21\}$.

3.1 Generating individual components

Owing to the different ways in which supposition configurations, interpretation graphs and explanatory extensions interact, we employ different techniques to generate each of these components: a dynamic priority queue is used to generate interpretation graphs; supposition configurations are drawn from a static pool based on a dynamic priority queue; and a deterministic algorithm is applied to generate explanatory extensions.

Generating interpretation graphs

A priority queue is initialized with the smallest BN subnet that connects a user's statements. An iterative process is then followed, where in each iteration, the candidate at the top of the queue is selected, its "children" are generated, and their probability is calculated (Section 4). A child of an interpretation graph is the smallest interpretation graph (BN subnet) that connects the argument propositions in a modified BN where an arc from the parent interpretation graph has been removed. For example, in Figure 1(b), the smallest interpretation graph that connects GreenInGardenAt11 with GreenHasOpportunity goes through GreenInGardenAtTimeOfDeath. If we remove the arc between GreenInGardenAt11 and GreenInGardenAtTimeOfDeath, BIAS generates an interpretation graph that goes from GreenInGardenAt11 to NbourHeardGreenBodyArgLastNight, GreenBodyArgueLastNight and GreenVisitBodyLastNight. The newly generated children are then slotted in the priority queue according to their probability. This process yields good results for interpretation graphs, as the order in which these graphs appear in the queue is indicative of their goodness (graphs that appear earlier are usually better).

Generating supposition configurations

Supposition configurations are generated by considering the following options for every node in the domain BN: (1) suppose nothing; (2) if the node is uninstantiated then instantiate it to TRUE and to FALSE; and (3) if the node is instantiated, uninstantiate it. The probability of each option is determined by its type (suppose nothing, instantiate or uninstantiate), and by how close a supposed belief is to the belief in the node in question, e.g., the probability of instantiating to TRUE a node with a belief of 0.8 is higher than the probability of instantiating it to FALSE. A

supposition configuration is generated by first taking the highest probability option for all the nodes (which is the "suppose nothing" option), then taking the next best option for each node in turn (leaving the others as they are), and so on. For instance, in our domain BN, the second most probable option consists of setting TimeOfDeath11 to TRUE without changing the beliefs in the other nodes, next setting just GreenVisitBodyLastNight to TRUE, and so on.

This process is based on the closeness between the (new) supposed beliefs and the system's beliefs obtained by Bayesian propagation from evidence. However, it does not take into account how well the system's resultant beliefs in the argument propositions match the user's stated beliefs. Hence, we cannot simply rely on supposition configurations that are generated early in the interpretation process, as later configurations may yield a better belief match overall. In order to be able to access these later candidates and still achieve close to anytime performance, we create a static pool of promising candidates at the start of the interpretation process. This pool is populated by calling a priority queue of supposition configurations M times as described above, and retaining the m best candidates ($M \gg m$) on the basis of both (1) how close are the supposed beliefs to the original beliefs in the BN, and (2) how close are the system's resultant beliefs in the argument propositions to those stated by the user (these factors are respectively related to model complexity and data fit, Section 4). During the interpretation process, a new supposition configuration is probabilistically selected from this pool (the priority queue is never recalled).

Generating explanatory extensions

We conducted surveys to assess the influence of the beliefs in the antecedents and consequent of probabilistic inferences on the acceptability of these inferences. The main insights from our surveys are that people object to two types of inferences: (1) those which have more certainty regarding the consequent than regarding the antecedents, e.g., *Probably A* $\Rightarrow$ ***Very*** *Probably C*; and (2) those where there is a large change in certainty from the antecedents to the consequent, e.g., *Probably A* $\Rightarrow$ *EvenChance C* [Zukerman and George 2005]. In addition, among *acceptable* inferences, people prefer *BothSides* inferences to *SameSide* inferences. *BothSides* inferences have antecedents with beliefs on both "sides" of the belief in the consequent (higher probability and lower), e.g., *A[VeryProbably] & B[PossiblyNot]* $\Rightarrow$ *C[Possibly]*; while all the antecedents in *SameSide* inferences have beliefs on "one side" of the belief in the consequent, e.g., *A[VeryProbably] & B[Probably]* $\Rightarrow$ *C[Possibly]* [George, Zukerman, and Niemann 2007].[6]

Explanatory extensions are generated by considering the siblings of the an-

[6]Our surveys were restricted to *direct* inferences, where a high/low probability for an antecedent yields a high/low probability for the consequent. We posit similar preferences for *inverse* inferences, where a low/high probability antecedent yields a high/low probability consequent. The work described in [George, Zukerman, and Niemann 2007] contains additional, more fine-grained categories of inferences, but here we restrict our discussion to the main ones.

tecedents of an unacceptable inference in an interpretation graph (these siblings are not in the graph), and assigning each sibling to an inference category according to its effect on the inference. The siblings in the most preferred inference category are then added to the interpretation graph as an explanatory extension, e.g., if there is a set of siblings that turns an unacceptable inference into a *BothSides* inference, then it is chosen. This simple approach yields interpretations that people find acceptable (Section 5). However, further investigation is required to determine whether different combinations of siblings would yield better results.

4 Probabilistic Formalism

As mentioned in Section 1, the Minimum Message Length (MML) principle [Wallace 2005] selects the simplest model that explains the observed data. In our case, the data are the argument given by a user, and the candidate models are interpretations of this argument. In addition to the data and the model, the MML principle requires the specification of background knowledge — information shared by the system and the user prior to the argument, e.g., domain knowledge (including shared beliefs) and dialogue history.

We posit that the best interpretation is that with the highest posterior probability.

$$IntBest = \text{argmax}_{i=1,\ldots,q} \Pr(IG_i, SC_i, EE_i | Argument, Background)$$

where q is the number of interpretations.

After assuming conditional independence between the argument and the background given an interpretation, this probability is represented as follows.

$$IntBest = \text{argmax}_{i=1,\ldots,q} \{\Pr(IG_i, SC_i, EE_i | Background) \times \Pr(Argument | IG_i, SC_i, EE_i)\} \quad (1)$$

The first factor, which is also known as *model complexity*, represents the prior probability of the model, and the second factor represents *data fit*.

- The **prior probability** of a model or **model complexity** reflects how "easy" it is to construct the model (interpretation) from background knowledge. For instance, complex models (e.g., interpretations with larger interpretation graphs) usually have a lower prior probability than simpler models.

- **Data fit** measures how similar the data (argument) are to the model (interpretation). The closer the data are to the model, the higher the probability of the data given the model (i.e., the probability that the user uttered the argument when he or she intended the interpretation in question).

Both the argument and its interpretation contain *structural* information and *beliefs*. The beliefs are simply those stated in the argument and in the interpretation, and suppositions made as part of the interpretation. The structural part of the

argument comprises the stated propositions and the relationships between them, while the structural part of the interpretation comprises the interpretation graph and explanatory extensions. As stated above, smaller, simpler structures usually have a higher prior probability than larger, more complex ones. However, the simplest structure is not necessarily the best overall. For instance, the simplest possible interpretation for any argument consists of a single proposition, but this interpretation usually yields a poor data fit with most arguments. An increase in structural complexity (and corresponding reduction in probability) may reduce the discrepancy between the argument structure and the structure of the interpretation graph, thereby improving data fit. If this improvement overcomes the reduction in probability due to the higher model complexity, we obtain a higher-probability interpretation overall.

The techniques employed to calculate the prior probability and data fit for both types of information are outlined below (a detailed description appears in [George, Zukerman, and Niemann 2007; Zukerman and George 2005]).

4.1 Prior probability of an interpretation: $\mathbf{Pr(IG_i, SC_i, EE_i | \textit{Background})}$

As mentioned above, the prior probability of an interpretation reflects how well it fits the background knowledge. In our case, the background knowledge comprises (1) domain knowledge – the evidence in the BN (known to the user and the system); (2) dialogue history – previously mentioned propositions; and (3) presentation preferences – features of acceptable inferences (obtained from user surveys).

To estimate the prior probability of an interpretation, we separate the structure of an interpretation and the beliefs in it (note that SC_i comprises only beliefs, and EE_i comprises only structure).

$$\begin{aligned}&\Pr(IG_i, SC_i, EE_i | Background) = \\ &\quad \Pr(\text{beliefs in } IG_i, \text{struct of } IG_i, SC_i, EE_i | Background)\end{aligned}$$

After applying the chain rule of probability, we obtain

$$\begin{aligned}&\Pr(IG_i, SC_i, EE_i | Background) = \\ &\quad \Pr(\text{beliefs in } IG_i | \text{struct of } IG_i, SC_i, EE_i, Background) \times \\ &\quad \Pr(EE_i | \text{struct of } IG_i, SC_i, Background) \times \\ &\quad \Pr(\text{struct of } IG_i | SC_i, Background) \times \Pr(SC_i | Background)\end{aligned} \tag{2}$$

The factors in Equation 2 are described below (we consider them from last to first for clarity of exposition).

Supposition configuration: $\mathbf{Pr(SC_i | \textit{Background})}$

A supposition configuration addresses mismatches between the beliefs expressed in an argument and those in an interpretation. It comprises beliefs attributed

to the user in light of the beliefs shared with the system, which are encoded in the background knowledge. Making suppositions has a lower probability than not making suppositions (which has no discrepancy with the background knowledge). However, as seen in the example in Figure 2(c), making a supposition that reduces or eliminates the discrepancy between the beliefs stated in the argument and those in the interpretation increases the data fit for beliefs.

$\Pr(SC_i|Background)$ reflects how close the suppositions in a supposition configuration are to the current beliefs in the background knowledge. The closer they are, the higher the probability of the supposition configuration. Assuming conditional independence between the supposition for each node given the background knowledge yields

$$\Pr(SC_i|Background) = \prod_{j=1}^{N} \Pr(s_{ji}|Bel_{Bkgrd}(j))$$

where N is the number of nodes in the BN, s_{ij} is the supposition made for node j in supposition configuration SC_i, and $Bel_{Bkgrd}(j)$ is the belief in node j according to the background knowledge. $\Pr(s_{ji}|Bel_{Bkgrd}(j))$ is estimated by means of the heuristic function $\mathcal{H}$.

$$\Pr(s_{ji}|Bel_{Bkgrd}(j)) = \mathcal{H}(Type(s_{ji}), Bel(s_{ji})|Bel_{Bkgrd}(j))$$

where $Type(s_{ji})$ is the type of supposition s_{ji} (supposing nothing, supposing evidence, or forgetting evidence), and $Bel(s_{ji})$ is the value of the supposition (TRUE or FALSE when evidence is supposed for node j; and the belief in node j obtained from belief propagation in the BN when evidence is forgotten for node j). Specifically, we posit that supposing nothing has the highest probability, and supposing the truth or falsehood of an inferred value is more probable than forgetting seen evidence [George, Zukerman, and Niemann 2007]. In addition, strongly believed (high probability) propositions are more likely to be supposed TRUE than weakly believed (lower probability) propositions, and weakly believed propositions are more likely to be supposed FALSE than strongly believed propositions [Lichtenstein, Fischhoff, and Phillips 1982].

Structure of an interpretation: Pr(struct of $\mathbf{IG_i}|\mathbf{SC_i}$, *Background*)

$\Pr(\text{struct of } IG_i|SC_i, Background)$ is the probability of selecting the nodes and arcs in IG_i from the domain BN (which is part of the background knowledge). The calculation of this probability is described in detail in [Zukerman and George 2005]. In brief, the prior probability of the structure of an interpretation graph is estimated using the combinatorial notion of selecting the nodes and arcs in the graph from those in the domain BN. To implement this idea, we specify an interpretation graph IG_i by indicating the number of nodes in it (n_i), the number of arcs (a_i), and the actual nodes and arcs in it ($Nodes_i$ and $Arcs_i$ respectively). Thus, the probability of the structure of IG_i in the context of the domain BN (composed of A arcs and

N nodes) is defined as follows.

$$\Pr(\text{struct } IG_i|Background) = \Pr(Arcs_i, Nodes_i, a_i, n_i|Background)$$

Applying the chain rule of probability yields

$$\begin{aligned}\Pr(\text{struct } IG_i|Background) \quad = \quad & \Pr(Arcs_i|Nodes_i, a_i, n_i, Background) \times \\ & \Pr(a_i|Nodes_i, n_i, Background) \times \\ & \Pr(Nodes_i|n_i, Background) \times \Pr(n_i|Background)\end{aligned} \tag{3}$$

These probabilities are calculated as follows.

- $\Pr(n_i|Background)$ is the probability of having n_i nodes in an interpretation graph. We model this probability by means of a truncated Poisson distribution, $Poisson(\beta)$, where β is the average number of nodes in an interpretation (obtained from user trials).

- $\Pr(Nodes_i|n_i, Background)$ is the probability of selecting the particular n_i nodes in $Nodes_i$ from the N nodes in the domain BN. The simplest calculation assumes that all nodes in an interpretation graph have an equal probability of being selected, i.e., there are $\binom{N}{n_i}$ ways to select these nodes. This calculation generally prefers small models to larger ones.[7] In [Zukerman and George 2005], we considered salience — obtained from dialogue history, which is part of the background knowledge — to moderate the probability of selecting a node. According to this scheme, recently mentioned nodes are more salient (and have a higher probability of being selected) than nodes mentioned less recently.

- $\Pr(a_i|Nodes_i, n_i, Background)$ is the probability of having a_i arcs in an interpretation graph. The number of arcs in an interpretation is between the minimum number of arcs needed to connect n_i nodes $(n_i - 1)$, and the actual number of arcs in the domain BN that connect the nodes in $Nodes_i$, denoted va_i. We model the probability of a_i by means of a uniform distribution between $n_i - 1$ and va_i.

- $\Pr(Arcs_i|Nodes_i, a_i, n_i, Background)$ is the probability of selecting the particular a_i arcs in $Arcs_i$ from the va_i arcs in the domain BN that connect the nodes in IG_i. Assuming an equiprobable distribution, there are $\binom{va_i}{a_i}$ ways to select these arcs.

Structure of an explanatory extension:
$\Pr(\mathbf{EE_i}|\text{struct of } \mathbf{IG_i}, \mathbf{SC_i}, \boldsymbol{Background})$

Explanatory extensions are added to an interpretation graph to accommodate people's expectations regarding the relationship between the antecedents of an inference

[7] In the rare cases where the number of propositions in an interpretation exceeds $N/2$, smaller models do not yield lower probabilities.

and its consequent (rather than to connect between the propositions in an argument). These expectations, which are part of the background knowledge, were obtained from our user studies. Explanatory extensions have no belief component, as the nodes in them do not provide additional evidence, and hence do not affect the beliefs in a BN.

Interpretations with explanatory extensions are more complex, and hence have a lower probability, than interpretations without such extensions. At the same time, as shown in the example in Figure 2(d), an explanatory extension that overcomes an expectation violation regarding the consequent of an inference improves the acceptance of the interpretation, thereby increasing the probability of the model.

According to our surveys, explanatory extensions that yield *BothSides* inferences are preferred to those that yield *SameSide* inferences. In addition, as for interpretation graphs, shorter explanatory extensions are preferred to longer ones. Thus, our estimate of the structural probability of explanatory extensions balances the size of explanatory extensions (*number of propositions*) against their type (*inference category*), as follows.[8]

$$\Pr(\text{struct of } EE_i|\text{struct of } IG_i, SC_i, \mathit{Background}) = \prod_{j=1}^{NF_i} \Pr(\mathit{InfCategory}(EE_{ij}), np(EE_{ij})|\text{struct of } IG_i, SC_i, \mathit{Background})$$

where NF_i is the number of inferences in IG_i, $\mathit{InfCategory}(EE_{ij})$ is the category of the inference obtained by adding explanatory extension EE_{ij} to the jth inference in IG_i, and $np(EE_{ij})$ is the number of propositions in EE_{ij}.

Applying the chain rule of probability yields

$$\Pr(\text{struct of } EE_i|\text{struct of } IG_i, SC_i, \mathit{Background}) = \prod_{j=1}^{NF_i} \left\{ \begin{array}{l} \Pr(\mathit{InfCategory}(EE_{ij})|np(EE_{ij}), \text{struct of } IG_i, SC_i, \mathit{Background}) \times \\ \Pr(np(EE_{ij})|\text{struct of } IG_i, SC_i, \mathit{Background}) \end{array} \right\} \quad (4)$$

These probabilities are calculated as follows.

- $\Pr(\mathit{InfCategory}(EE_{ij})|np(EE_{ij}), \text{struct of } IG_i, SC_i, \mathit{Background})$ is estimated using a heuristic function that represents people's preferences: an explanatory extension that yields a *BothSides* inference has a higher probability than an explanatory extension that yields a *SameSide* inference.

- As for interpretation graphs, $\Pr(np(EE_{ij})|\text{struct of } IG_i, SC_i, \mathit{Background})$ is estimated by means of a truncated Poisson distribution, $\mathit{Poisson}(\mu)$, where μ is the average number of nodes in an explanatory extension.

[8] We do not estimate the probability of including particular nodes in an explanatory extension, because the nodes in an explanatory extension are completely determined by their inference category.

Beliefs in an interpretation:
Pr(beliefs in $\mathbf{IG_i}$|struct of $\mathbf{IG_i}, \mathbf{SC_i}, \mathbf{EE_i}$, *Background*)

The beliefs in an interpretation IG_i are estimated by performing Bayesian propagation from the beliefs in the domain BN and the suppositions. This is an algorithmic process, hence the probability of obtaining the beliefs in IG_i is 1. However, the background knowledge has another aspect, viz users' expectations regarding inferred beliefs. In our preliminary trials, users objected to inferences that had increases in certainty or large changes in belief from their antecedents to their consequent [Zukerman and George 2005].

Thus, interpretations that contain objectionable inferences have a lower probability than interpretations where the beliefs in the consequents of the inferences fall within an "acceptable range" of the beliefs in their antecedents. We use the categories of acceptable inferences obtained from our surveys to estimate the probability of each inference in an interpretation — these categories define an *acceptable range* of beliefs for the consequent of an inference given its antecedents. For example, an inference with antecedents *A[Probably] & B[Possibly]* has the acceptable belief range {*Probably, Possibly, EvenChance*} for its consequent. The probability of an inference whose consequent falls within the acceptable range is higher than the probability of an inference whose consequent falls outside this range. In addition, we extrapolate from the results of our surveys, and posit that the probability of an unacceptable inference decreases as the distance of its consequent from the acceptable range increases. We use the Zipf distribution to model the probability of an inference, where the "rank" is the distance between the belief in the consequent and the acceptable belief range.

As mentioned above, explanatory extensions are generated to satisfy people's expectations about the relationship between the beliefs in the antecedents of inferences and the belief in their consequent (i.e., they bring the consequent into the acceptable range of an inference, or at least closer to this range). Thus, they increase the belief probability of an interpretation at the expense of its structural probability.

4.2 Data fit between argument and interpretation: Pr(*Argument*|$\mathbf{IG_i}, \mathbf{SC_i}, \mathbf{EE_i}$)

As mentioned above, data fit reflects the probability that a user who intended a particular interpretation generated the given argument. This probability is a function of the similarity between the argument and the interpretation: the higher this similarity, the higher the probability of the argument given the interpretation. As for prior probabilities, we consider structural similarity and belief similarity.

$$\begin{aligned}\Pr(\mathit{Argument}|IG_i, SC_i, EE_i) \quad = \quad & \Pr(\text{struct of } \mathit{Argument}, \text{ beliefs in } \mathit{Argument}| \\ & \text{struct of } IG_i, \text{ beliefs in } IG_i, SC_i, EE_i)\end{aligned}$$

We assume that given an interpretation graph, the argument is independent of

the suppositions and explanatory extensions, which yields

$$\begin{aligned} \Pr(Argument|IG_i, SC_i, EE_i) \quad = \quad & \Pr(\text{struct of } Argument|\text{struct of } IG_i) \times \\ & \Pr(\text{beliefs in } Argument|\text{beliefs in } IG_i) \end{aligned} \tag{5}$$

Structure of the argument given the structure of an interpretation: Pr(struct of *Argument*|struct of $\mathbf{IG_i}$)

The estimation of the structural similarity between an argument and an interpretation is based on the idea that the nodes and arcs in the argument are selected from those in the interpretation graph. This idea is similar to that used to calculate the prior probability of the structure of the interpretation graph, where the nodes and arcs in IG_i were selected from those in the domain BN (Section 4.1). However, in this case there is a complicating factor, since as seen in our examples, a user may mention implications (arcs) which are absent from the interpretation graph (our web interface prevents the inclusion of nodes that do not appear in the domain BN). Hence, the calculation of Pr(struct of *Argument*|struct of IG_i) is similar to the calculation of Pr(struct of $IG_i|Background$) in Equation 3, but it distinguishes between arcs in *Argument* that are selected from IG_i and arcs that are newly inserted.

Beliefs in the argument given the beliefs in an interpretation: Pr(beliefs in *Argument*|beliefs in $\mathbf{IG_i}$)

The closer the beliefs stated in an argument are to the beliefs in an interpretation, the higher the probability that the user presented the argument when he or she intended this interpretation. Thus, suppositions that reduce belief discrepancies between the beliefs in an argument and those in an interpretation improve data fit (at the expense of the prior probability of the interpretation, Section 4.1). We employ the Zipf distribution to estimate the probability that the beliefs in an interpretation were intended in the argument, where the "rank" is the difference between the corresponding beliefs.

In a final step, we inspect the interpretation graph to determine whether it has blocked paths. The presence of blocked paths in an interpretation graph suggests that the lines of reasoning in the interpretation do not match those in the argument. Thus, if blocked paths are found, the probability of the belief match between the interpretation graph and the argument is reduced.

5 Evaluation

Our evaluation was designed to determine whether our approach to argument interpretation by a Bayesian intelligence yields interpretations that are acceptable to users. However, our target users are not those who constructed the argument, but those who read the argument. Specifically, our evaluation determines whether people reading someone else's argument find BIAS's highest-probability interpretations acceptable (and better than other options).

We evaluated the components of an interpretation separately in the order in which they were developed: first interpretation graphs, next supposition configurations, and finally explanatory extensions. Our evaluations relied on scenarios based on BNs that were similar to that in Figure 1(b). Our participants were staff and students in the Faculty of Information Technology at Monash University and people known to the project team members (the participants exhibited different levels of computer literacy).

Interpretation graphs. We constructed four evaluation sets, where each set contained an argument (the argument in Figure 1(a), and three short arguments similar to that in Figure 2(a)) and BIAS's preferred interpretations.[9] Between 17-25 participants read each argument and the interpretations. They were then asked to give each interpretation a score between 1 (Very UNreasonable) and 5 (Very Reasonable), and to comment on aspects of the interpretations that they liked or disliked. People generally found our interpretations acceptable, with average scores between 3.35 and 4. The lower scores were attributed to three main problems: (1) participants' disagreement with the systems' domain-related inferences, (2) discrepancies between the argument's beliefs in the consequents of inferences and the system's beliefs, and (3) unacceptable inferences. The first problem is discussed in Section 6, and the other two problems are addressed by supposition configurations and explanatory extensions respectively.

Supposition configurations. We constructed four scenarios, where each scenario had two versions of an argument: (1) an original version, whose beliefs were obtained by Bayesian propagation in the domain BN; and (2) a version given by a hypothetical user, whose conclusion did not match that of the original version (BIAS had to make a supposition in order to account for the beliefs in the user's argument). 34 participants read both arguments, and were then asked whether it was reasonable to make suppositions about the user's beliefs in order to make sense of his or her argument. To answer this question, they could (1) select one of four suppositions we showed them (which included BIAS's top-ranked supposition and other highly ranked suppositions), (2) include an alternative supposition of their choice (from BIAS's knowledge base or of their own devising), or (3) indicate that no suppositions were required. BIAS's preferred supposition was consistently ranked first or second by our trial subjects, with its average rank being the lowest (best) among all the options. In addition, very few respondents felt that no suppositions were warranted.

Explanatory extensions. We constructed two evaluation sets, each consisting of a short argument and two alternative interpretations — one with explanatory extensions and one without. These sets were shown to 20 participants. The majority of the participants preferred the interpretations with explanatory extensions. At

[9]The arguments were generated by project team members. We also conducted experiments were people not associated with the project entered arguments, but interface problems affected the evaluation (Section 6.4).

the same time, about half of the participants felt that the extended interpretations were too verbose. This problem may be partially attributed to the presentation of the nodes as direct renditions of their propositional content, which makes the interpretations appear repetitive in style. The generation of stylistically diverse text is the subject of active research in Natural Language Generation, e.g., [Gardent and Kow 2005].

6 Discussion

This chapter offers a probabilistic approach to argument interpretation by a system that uses a BN as its knowledge representation and reasoning formalism. An interpretation of a user's argument is represented as beliefs in the BN (suppositions) and a Bayesian subnet (interpretation graph and explanatory extensions). Our evaluations show that people found BIAS's interpretations generally acceptable, and its suppositions and explanatory extensions both necessary and reasonable.

Our approach casts the generation of an interpretation as a model selection task, and employs an (almost) anytime algorithm to generate candidate interpretations. Our model selection approach balances the probability of the model in light of background knowledge against its data fit (similarity between the model and the data). In other words, our formalism balances the cost of adding extra elements to an interpretation (e.g., suppositions) against the benefits obtained from these elements. The calculations that implement this idea are based on three main elements: (1) combinatoric principles for extracting an interpretation graph from the domain BN, and an argument from an interpretation; (2) known distributions, such as Poisson for the number of nodes in an interpretation graph or explanatory extension, and Zipf for modeling discrepancies in belief; and (3) manually-generated distributions for suppositions and for preferences regarding different types of inferences. The parameterization of these distributions requires specific information. For instance, the mean of the Poisson distribution, which determines the "penalty" for having too many nodes in an interpretation or explanatory extension, must be empirically determined. Similarly, the hand-tailored distributions for supposition configurations and explanatory extensions require experimental fine-tuning or user studies to gather these probabilities.

The applicability of our approach is mainly affected by our assumption that the nodes in the domain BN are binary. Other factors to be considered when applying our formalism are: the characteristics of the domain, the expressive power of BNs *vis a vis* human reasoning, and the ability of users to interact with the system.

6.1 Binary node BNs

The assumption that the nodes in the domain BN are binary simplifies the estimation of the probability of suppositions and explanatory extensions. The relaxation of this assumption to multi-valued nodes would increase the search space for suppositions, and necessitate a generalization of the heuristics used to calculate the

probability of a supposition (Section 4.1). The incorporation of multi-valued nodes would also require a generalization of the procedure for generating explanatory extensions and estimating their probability (Sections 3.1 and 4.1 respectively). This in turn would necessitate user studies to determine people's presentation preferences and expectations about inferences involving multi-valued nodes. For instance, people may prefer such inferences to be presented in terms of a particular value of a node or in terms of an aggregate of several values; and different models of expectations may be required for nodes with ordinal values (e.g., low, medium and high) and nodes with scalar values (e.g., colours). Further, as indicated in Section 6.2, these preferences and expectations are domain dependent.

6.2 Domain of argumentation

We selected a "commonsense" domain both for ease of design and to be able to conduct trials with non-experts. The nodes and arcs in the domain BN and the values in the Conditional Probability Tables (CPTs) were devised by the project team members. A consequence of working in a commonsense domain is that people, rather than computer systems, are the domain experts. As a result, users may postulate ideas of which the system is unaware (e.g., Mr Green and Ms Scarlet were in cahoots), and their inferences may validly differ from those of the system. For instance, according to BIAS, Mr Green and Mr Body being enemies implies that Mr Green very probably has a motive to kill Mr Body — an inference that several users found objectionable.

To deal with the first of these issues, an argumentation system can (1) restrict the user to use only the propositions known to the system, (2) ignore the user's propositions that are not known to the system, or (3) try to learn the import of new propositions. Our experience with BIAS shows that the first solution is frustrating for users, as people did not like having to shoehorn their reasoning into the propositions known to the system. The second solution leads to only a partial understanding of the user's intentions, and hence potentially to a mis-directed discussion. The third solution, which also applies to the synchronization of inference patterns between the user and the system, is clearly the most sound. However, incorporating new propositions into a BN, and modifying inference patterns, have significant implications with respect to the system's reasoning, and present non-trivial interface design problems. These observations, together with the fact that at present the strength of computer systems is their ability to perform expert reasoning, indicate that a fruitful domain for the incorporation of argumentation capabilities into BNs is an expert domain, where the system's knowledge generally exceeds that of users.

Our procedures for generating interpretation graphs and supposition configurations are domain independent. However, the generation of explanatory extensions is domain dependent. This is because explanatory extensions are generated to explain surprising outcomes, and what is surprising often depends on the domain. Further, in some domains what matters is the increase or reduction in probability,

rather than its absolute value, e.g., an increase from 6% to 10% in the probability of a patient having cancer may require an explanatory extension, even though both probabilities belong to the *VeryProbablyNot* belief category.

6.3 BN reasoning and human reasoning

Our approach assumes that the underlying BN represents only ground predicates, while human reasoning often involves general statements (quantified predicates). Getoor *et al.* [Getoor, Friedman, Koller, and Taskar 2001] and Taskar *et al.* [Taskar, Abbeel, and Koller 2002] studied probabilistic relational models, which combine advantages of relational logic and BNs, and can generalize over a variety of situations. This is a promising representation for the interpretation of arguments that include quantified predicates.

Belief propagation in BNs differs from human belief propagation when users employ different inference patterns from those in the BN, and when users do not engage in normative probabilistic reasoning. As mentioned above, the synchronization of inference patterns between the user and the system is a challenging task which falls under the purview of probabilistic reasoning and human-computer interfaces.

People's non-normative probabilistic reasoning is partly attributed to *reasoning fallacies* [Evans, Barston, and Pollard 1983; Lichtenstein, Fischhoff, and Phillips 1982; Tversky and Kahneman 1982]. In previous research, we augmented a Bayesian argument generation system with a (rather coarse) model of certain types of human reasoning fallacies [Korb, McConachy, and Zukerman 1997]. An interesting avenue for future research consists of developing finer, domain dependent models of human reasoning fallacies, and incorporating them into our interpretation process.

6.4 Argumentation interface

BIAS requires users to construct their arguments using only propositions known to the system, and assumes that the arguments are in premise-to-goal form. As mentioned in Section 6.2, users disliked having to shoehorn their ideas into a restricted set of propositions. An alternative approach, which we considered in [Zukerman, George, and Wen 2003], allowed users to provide Natural Language statements, and then mapped these statements to propositions in the system's knowledge base. However, such a process runs the risk of producing an erroneous mapping. Hence, this process should be able to determine when a mapping is questionable, and handle this situation appropriately.

In addition to a premise-to-goal argumentation strategy, people employ strategies such as reductio-ad-absurdum, inference to best explanation, and reasoning by cases [Zukerman, McConachy, and Korb 2000]. These strategies must be identified prior to rendering an argument into an interpretation graph. An interesting approach for addressing this problem involves using a graphical interface to help users structure an argument [van Gelder 2005], while allowing them to express propositions in Natural Language.

Acknowledgments

The author thanks her collaborators on the research described in this chapter: Sarah George and Michael Niemann. This research was supported in part by grant DP0878195 from the Australian Research Council (ARC) and by the ARC Centre for Perceptive and Intelligent Machines in Complex Environments.

References

Dean, T. and M. Boddy (1988). An analysis of time-dependent planning. In *AAAI88 – Proceedings of the 7th National Conference on Artificial Intelligence*, St. Paul, Minnesota, pp. 49–54.

Elsaesser, C. (1987). Explanation of probabilistic inference for decision support systems. In *Proceedings of the AAAI-87 Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington, pp. 394–403.

Evans, J., J. Barston, and P. Pollard (1983). On the conflict between logic and belief in syllogistic reasoning. *Memory and Cognition 11*, 295–306.

Gardent, C. and E. Kow (2005). Generating and selecting grammatical paraphrases. In *ENLG-05 – Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, Scotland, pp. 49–57.

George, S., I. Zukerman, and M. Niemann (2007). Inferences, suppositions and explanatory extensionsin argument interpretation. *User Modeling and User-Adapted Interaction 17*(5), 439–474.

Getoor, L., N. Friedman, D. Koller, and B. Taskar (2001). Learning probabilistic models of relational structure. In *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, Massachusetts, pp. 170–177.

Horvitz, E., H. Suermondt, and G. Cooper (1989). Bounded conditioning: flexible inference for decision under scarce resources. In *UAI89 – Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence*, Windsor, Canada, pp. 182–193.

Kashihara, A., T. Hirashima, and J. Toyoda (1995). A cognitive load application in tutoring. *User Modeling and User-Adapted Interaction 4*(4), 279–303.

Kintsch, W. (1994). Text comprehension, memory and learning. *American Psychologist 49*(4), 294–303.

Korb, K. B., R. McConachy, and I. Zukerman (1997). A cognitive model of argumentation. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, Stanford, California, pp. 400–405.

Lichtenstein, S., B. Fischhoff, and L. Phillips (1982). Calibrations of probabilities: The state of the art to 1980. In D. Kahneman, P. Slovic, and A. Tversky (Eds.), *Judgment under Uncertainty: Heuristics and Biases*, pp. 306–334. Cambridge University Press.

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

Taskar, B., P. Abbeel, and D. Koller (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, Alberta, Canada, pp. 485–490.

Tversky, A. and D. Kahneman (1982). Evidential impact of base rates. In D. Kahneman, P. Slovic, and A. Tversky (Eds.), *Judgment under Uncertainty: Heuristics and Biases*, pp. 153–160. Cambridge University Press.

van Gelder, T. (2005). Teaching critical thinking: some lessons from cognitive science. *College Teaching 45*(1), 1–6.

Wallace, C. (2005). *Statistical and Inductive Inference by Minimum Message Length*. Berlin, Germany: Springer.

Zukerman, I. and S. George (2005). A probabilistic approach for argument interpretation. *User Modeling and User-Adapted Interaction 15*(1-2), 5–53.

Zukerman, I., S. George, and Y. Wen (2003). Lexical paraphrasing for document retrieval and node identification. In *IWP2003 – Proceedings of the 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications*, Sapporo, Japan, pp. 94–101.

Zukerman, I., R. McConachy, and K. B. Korb (1998). Bayesian reasoning in an abductive mechanism for argument generation and analysis. In *AAAI98 – Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, Wisconsin, pp. 833–838.

Zukerman, I., R. McConachy, and K. B. Korb (2000). Using argumentation strategies in automated argument generation. In *INLG'2000 – Proceedings of the 1st International Conference on Natural Language Generation*, Mitzpe Ramon, Israel, pp. 55–62.