Graphical models for problem solving

Article *in* Computing in Science and Engineering · August 2000

DOI: 10.1109/5992.852390 · Source: IEEE Xplore

3 authors:

Carlos Alberola-López
Universidad de Valladolid
240 PUBLICATIONS **2,833** CITATIONS
SEE PROFILE

Lorenzo J. Tardon
University of Malaga
87 PUBLICATIONS **421** CITATIONS
SEE PROFILE

Juan Ruiz-Alzola
Universidad de Las Palmas de Gran Canaria
94 PUBLICATIONS **1,229** CITATIONS
SEE PROFILE

GRAPHICAL MODELS FOR PROBLEM SOLVING

Graphical models are not just efficient computational structures—they also incorporate knowledge in an intuitive way. The authors give an overview of the most important graphical models and some interesting applications of each.

People tend naturally to resort to graphs when solving a problem, if only to more fully represent that problem. Just think of a technical presentation without charts, graphs, or the like. The audience would have trouble keeping track of the speaker's information, and the speaker wouldn't easily get to the point where his or her ideas were structured so as to make them understandable to others.

Think now of a complex problem in which a large number of variables interact. We know that some of these variables affect each other more than others; we even have some idea how some of them shield each other from the rest of the problem's variables. A graphical model is a natural way for us to visually capture this interaction and analyze it.

This article deals with *probabilistic systems*—systems in which uncertainty is present because the variables are random, so we can only draw probabilistic conclusions. The basic idea underlying *graphical models* is the exploitation of our ability to model complex problems out of small interacting boxes. However, quantifying this—associating numbers with the connections so as to make a consistent probabilistic system—is not trivial. The converse is also true: inferring a probability model from a graphical model is not obvious. Trying to verify the existence of a graphical model that captures all the dependence relations within a probabilistic system is a risky but potentially rewarding business. Here, we overview graphical models and provide applications and details to help researchers understand them.

1521-9615/00/$10.00 © 2000 IEEE

CARLOS ALBEROLA
Universidad de Valladolid
LORENZO TARDÓN
University of Malaga
JUAN RUIZ-ALZOLA
Universidad de Las Palmas de Gran Canaria

An axiomatic theory

Graphical modeling is a natural way for people to encode the global interaction among a problem's variables. (See the sidebar "A note on history" for more background.) We build such models using two basic elements: *nodes*, which encode the variables, and *connections*, which encode the interactions. Directly connected nodes exert direct influence. Indirectly connected nodes influence each other but only through the intermediate nodes that separate them.

Can a graphical model represent a probabilistic system—that is, a function P of, say, N variables $X_1, \ldots, X_N$ that satisfies all the axioms of probability?[1] Such functions involve marginalization, conditional probabilities, independencies (either

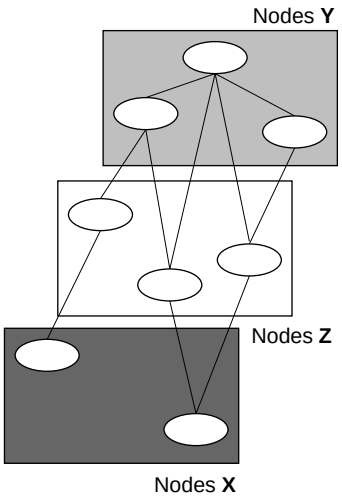

Figure 1. Separation in graphs.

marginal or conditional), and other issues.

Judea Pearl[1] approached this problem by defining a general dependency relation I on three sets of variables **X**, **Z**, and **Y**. (We use boldface to denote a set of variables and regular italic font to denote an individual variable. For instance, **X** might consist of N nodes (variables) $X_1, \ldots, X_N$. The relation I(**X**, **Z**, **Y**) is to be interpreted as "given the piece of knowledge **Z**, getting to know **Y** has no effect on our current belief of **X.**" A probabilistic model is but one particular case of such a general relation, but if relation I is a dependence relation with respect to a probability function P, this relation must satisfy a number of properties (see the "Pearl's theoretical framework" sidebar). The properties in the sidebar are the starting point to answer the question we have just posed.

Assume that **X**, **Z**, and **Y** are three sets of disjoint vertices in a graph. Common sense dictates (and we will take it for granted from now on) that the set **Z** separates the sets **X** and **Y** if every path that joins any two vertices of the two sets passes through vertices of **Z** (see Figure 1). In this case, **Z** is called a *cutset* of **X** and **Y**; if we now add vertices to **Z** (not belonging either to **X** or **Y**), the new superset keeps its cutset condition. This is actually a property called *strong union*. But, as you can see in the sidebar, I_P only needs to satisfy the weak union property; therefore, because weak union is weaker than node separation, we might find cases where graphical models cannot represent probabilistic models. We try to clarify this later, but first, we describe a broad classification of graphical models.

For a graphical model consisting of nodes and connections, the taxonomy is straightforward:

- If connections are undirected, we have an *undirected graph* (UG).
- If connections are directional (an arrow is drawn from an *origin* node to a different *destination* node), the graph is known as a *directed acyclic graph* (DAG).

The presence of arrows in a DAG forces a cause–consequence relation between nodes. Ori-

A note on history

Sewal Wright[1] is considered the pioneer in the use of graphical models for probabilistic information as an aid in the biometric analysis of data; his first work was published in 1921. However, statisticians in the first half of the 20th century mainly focused on quantitative analysis, so Wright's idea had a lukewarm acceptance.

The 1960s reversed that trend: people used graphical models to decompose statistical tables, the properties of these models started to catch on, and valuable contributions have appeared since then.[2–4]

Ernst Ising[5] pioneered the evolution of Markov random fields (MRFs) with his attempt to obtain a probabilistic model of the spin configurations of the particles belonging to a ferromagnetic material. He assumed that only neighboring particles influenced others and that the influence was one-dimensional.

Charles J. Preston generalized these results from regular topologies in a plane to arbitrary interactions, carefully defining the concepts of neighbors and cliques.[6] Julian Besag developed a general form to define MRFs in lattices by introducing the concept of potentials.[7] Finally, Stuart Geman and Donald Geman[8] proved a number of results about convergence of Markov chains. They showed how to use the Gibbs sampler and simulated annealing to solve problems posed as MRFs.

References

1. S. Wright, "Correlation and Causation," *J. Agricultural Research*, Vol. 20, 1921, pp. 557–585.
2. J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco, Calif., 1988.
3. D. Spiegelhalter, "Probabilistic Reasoning in Predictive Expert Systems," *Uncertainty in Artificial Intelligence*, L.N. Kanal and F. Lemmer, eds., North-Holland, Amsterdam, 1986, pp. 47–68.
4. S. Lauritzen and D. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems," *J. Royal Statistical Soc., Series B*, Vol. 50, No. 2, 1998, pp. 157–224.
5. R. Kinderman and J.L. Snell, *Markov Random Fields and their Applications*, American Mathematical Soc., Providence, R.I., 1980.
6. C. J. Preston, *Gibbs States in Countable Sets*, Cambridge Univ. Press, Cambridge, UK, 1974.
7. J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems," *J. Royal Statistical Soc., Series B*, Vol. 34, 1974, pp. 192–236.
8. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 6, Nov. 1984, pp. 721–741.

Pearl's theoretical framework

Three sets of theorems, taken directly from Judea Pearl's work,[1] give an objective framework to connect the conditional-independence relation with probability theory and to define clearly when undirected graphs and directed acyclic graphs are perfect maps of a probability function. We can understand conditional independency in a broader sense than what probability says. For three disjoint sets of variables **X**, **Y**, and **Z** (all belonging to the universal set **U**), the independence of **X** and **Y** conditioned to **Z** is to be understood as "once I get to know **Z**, knowing **Y** gives no further information about **X**." Probability theory is just a particular case.

In the following, the symbol $\cup$ represents the union of sets of variables—the superset containing all the nodes belonging to both operands. Also, Greek lower-case letters denote single nodes, and bold-face capital letters denote groups of nodes. Also, the & stands for AND.

Theorem 1

Given the disjoint sets of variables **X**, **Y**, and **Z**, if relation $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ means "**X** is independent of **Y** given **Z**" in some probabilistic model *P*, then the following conditions must be satisfied:

- Symmetry: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) - I_P(\mathbf{Y}, \mathbf{Z}, \mathbf{X})$
- Decomposition: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ & $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{W})$
- Weak union: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y})$
- Contraction: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ & $I_P(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$
- Intersection, if *P* is strictly positive: $I_P(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y})$ & $I_P(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$

Theorem 2

A necessary and sufficient condition for a probabilistic model *P* to be graph-isomorphic is that I_P must satisfy the following additional conditions:

- Strong union: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z} \cup \mathbf{W}, \mathbf{Y})$
- Transitivity: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \gamma)$ OR $I_P(\gamma, \mathbf{Z}, \mathbf{Y})$, with γ a single node that belongs to **U**, and all the arguments in $I_P(,,)$ being disjoint
- Intersection, to be satisfied always

Theorem 3

A necessary and sufficient condition for a probabilistic model *P* to be DAG-isomorphic is that I_P must satisfy the following additional conditions (with respect to Theorem 1):

- Composition/decomposition: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) - I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ & $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{W})$
- Weak transitivity: $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ & $I_P(\mathbf{X}, \mathbf{Z} \cup \gamma, \mathbf{Y}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \gamma)$ OR $I_P(\gamma, \mathbf{Z}, \mathbf{Y})$
- Chordality: $I_P(\alpha, \gamma \cup \delta, \beta)$ & $I_P(\gamma, \alpha \cup \beta, \delta) \Rightarrow I_P(\alpha, \gamma, \beta)$ OR $I_P(\alpha, \delta, \beta)$

Reference

1. J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco, Calif., 1988.

gin nodes are considered direct causes of destination nodes, and the latter are considered direct consequences of the former. Therefore, DAGs can encode knowledge in which a clear direction of dependence exists.

UGs encode *influences* among nodes; directly connected nodes have a stronger influence than vertices separated by intermediate nodes. In fact, as we said before, if a group of nodes are a cutset between two different sets of nodes, then knowing the values of the cutset will render one of the groups irrelevant to the other.

Undirected graphs

Consider the following problem. You are driving and see a policeman performing a speed control (monitoring traffic using radar). You know the police perform speed controls the day before a holiday (because many cars are on the road) or if there is a party nearby (for safety reasons). Assume you live in a college town—it's the summer term, so parties occur regardless of tomorrow being a workday. We can therefore assume independence between the holiday and the party. Initially, we can model this problem (see Figure 2) with three vertices: speed control (SC), party (F, for the Spanish word *fiesta*), and holiday (H). There is an obvious influence of party and holiday on speed control, so connections are drawn accordingly. But, because we have assumed F and H are independent, no connection appears between them.

This simple example highlights an important pitfall of UGs: the inability to model *induced dependencies*. The model separates the nodes F and H by means of SC, which is therefore the cutset of both nodes. Hence, according to the model, H and F are still independent—learning anything about one of them has no effect on the other. However, if I know tomorrow is not a holiday, the presence of the police car performing speed control must be because a party is happening somewhere. SC cannot be a cutset.

Another possibility is to join each and every vertex, but that would make Holiday and Party

initially dependent, which is false under our assumptions.

Thus, this example illustrates that UGs cannot encode all probabilistic systems. To make this statement formal, let G be a graph consisting of the vertices **V** and the edges **E**, or $G = (\mathbf{V}, \mathbf{E})$. Let a probabilistic system P consist of the variables **U** with three disjoint subsets **X**, **Z**, and **Y**. Assume we can draw a graph with as many vertices as variables in **U**. Therefore, think of variables in **U** and vertices in **V** interchangeably.

We make the following distinction:

- *Separation in a graph*: if **Z** is a cutset in a graph of the two sets **X** and **Y**, we represent it by $Sep(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$.
- *Conditional independence*: if **X** is conditionally independent of **Y** given **Z** (that is, if $P(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) = P(\mathbf{X} \mid \mathbf{Z})P(\mathbf{Y} \mid \mathbf{Z})$), then we write $I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$.

For the graph $G = (\mathbf{V}, \mathbf{E})$ to fully represent a probability model, separation in the graph must imply conditional independence and vice versa. In this case, the probabilistic model P is said to be *graph-isomorphic*, and the graph is said to be a perfect map of the probabilistic system. The "Pearl's theoretical framework" sidebar shows the conditions that the probability system must hold to be graph-isomorphic.

In the speed control example just given, separation in the graph obviously does not imply conditional independence; this is why the UG failed to represent such dependency. The property of *strong union*, for instance, does not hold (consider when $\mathbf{Z} = \emptyset$ and $\mathbf{W} = SC$).

In such cases, a perfect map is not possible, but sometimes an approximation to it is. This approximation is called an *independency map*, or I-map—a graph in which nodes separated by a cutset are conditionally independent but not all conditionally independent nodes are necessarily graph-separated.

We now face two issues with practical implications:[1]

- Given a probabilistic system P, can we find at least an I-map G of P?
- Given a graph G, can we build a probabilistic system P that is at least an I-map of G?

The answer to both questions is yes if the probabilistic system is *strictly positive*; that is, if all the events you can think of concerning the variables involved have a non-null probability.

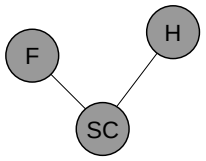

Figure 2. A simple undirected graph modeling a speed-control problem.

We can build an I-map of P in a few ways. First, we can start with a complete graph (all nodes are connected to each other). Delete those connections between two nodes, say A and B, if $I_P(A, \mathbf{U} - A - B, B)$—that is, if the two nodes are conditionally independent given the rest of the nodes. In this case, the I-map will have a minimum number of edges. Such a map is called a *Markov network* of P.

The second method is probably more intuitive to those familiar with Markov models because it makes use of the concept of a *neighbor system*. As is well known, a Markov model is characterized by the fact that knowing the current state of a number of variables *wrapping* the variable of interest (see Figure 3) renders the rest of the graph's variables irrelevant. In our terminology, denoting the neighbors (wrappers) of node A as $\mathbf{B}(A)$, then $I_P(A, \mathbf{B}(A), \mathbf{U} - \mathbf{B}(A) - A)$ holds.

These ideas lead to a simpler method: Start with as many nodes as variables in **U**, and connect every variable A to all its neighbors (all the nodes in $\mathbf{B}(A)$). The resulting graph is the same as the one we would get from using edge deletion.

J.M. Hammersley and P. Clifford answered the second question we posed earlier in 1971 in what was called thereafter the Hammersley-Clifford theorem.[2] It states that for a given UG, any function formed by a normalized product of strictly positive functions of the UG's cliques (subgraphs in which nodes are all adjacent to each other) is the probability function of a *Markov random field* relative to the graph.

Actually, it is possible to show[3] that we can express an MRF's probability function as a Gibbs function

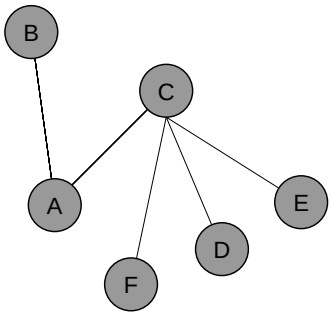

Figure 3. A tree-structured undirected graph.

The Ising model

The original formulation given by Ernst Ising[1] in his pioneering work starts by considering a sequence of points in the line 0, 1,. . ., n. Each point, in his ferromagnetic context, can have a "spin" up or down. We assign a probability measure $p(\Omega)$ on the set of all possible configurations of spins $\omega_l \in \Omega$ with $\Omega = (\omega_0, \omega_1, \ldots, \omega_n)$.

Let $\sigma(\omega_j) = \pm 1$ for up and down spin at node j. An energy $U(\Omega)$ is assigned to each configuration:

$$U(\Omega) = -J \sum_{i-j=1} \sigma(\omega_i)\sigma(\omega_j) \quad \text{(A)}$$

where J is a constant associated to a specific material. Ising made the simplifying assumption that only the interaction between neighboring nodes needs to be considered.

The next step assigns probabilities to the configurations Ω proportional to

$$e^{-\frac{1}{kT}U(\Omega)} \quad \text{(B)}$$

where T is temperature and k is the Boltzman constant. Finally,

$$p(\Omega) = \frac{1}{Z} e^{-\frac{1}{kT}U(\Omega)}, \quad Z = \sum_{\Omega} e^{-\frac{1}{kT}U(\Omega)} \quad \text{(C)}$$

A probability measure of the form in Equation C is called a *Gibbs measure*.

After that, the most important characterization from the point of view of probability arises—namely, the Markovian property:

$$p(\omega_j \mid \omega_k, k \neq j) = p(\omega_j \mid \omega_k, k \in N_j) \quad \text{(D)}$$

where N_j is the neighborhood of the node j (the nodes $j + 1$ and $j - 1$ in this formulation). A probability measure with the property in Equation D is called a *Markov random field*.

Reference

1. R. Kinderman and J.L. Snell, *Markov Random Fields and Their Applications*, American Mathematical Soc., Providence, R.I., 1980.

$$\Pi(\mathbf{X}) = \frac{e^{-H(\mathbf{X})}}{\sum_Z e^{-H(\mathbf{Z})}} \quad (1)$$

where the energy function H (see "The Ising model" sidebar) must be strictly positive. The denominator is just the normalizing constant required in the theorem. We can also see that if A is a node belonging to $\mathbf{X}$, then

$$P(A \mid \mathbf{X}) = P(A \mid \mathbf{B}(A)). \quad (2)$$

This is just the Markovian property. Far less obvious is the fact that this is a function of cliques with nodes belonging only to $A \cup \mathbf{B}(A)$.

Two implications of the theorem follow naturally. First, simple as it is, the theorem acts as a bridge between common sense and consistency, because it lets us encode local interactions as desired in a consistent, probabilistic system. For example, consider a landscape image that is blurred due to motion in the imaging process. To restore the degraded image by means of a probabilistic technique, we need an a priori probability model of the image.

The image consists of several homogeneous regions (a forest and a river) with sharp transitions only in the boundaries between the forest and the water. Such an image is characterized by smoothness; only a small percentage of pixels show sharp transitions. Consequently, a probabilistic model that tried to approximate such behavior should consist of decreasing functions of the cliques, thus giving low probability values when cliques show large differences and high probabilities when they show large similarities. A tentative energy function with this behavior is

$$H(\mathbf{X}) = \beta \sum_{<s,t>} (A^s - B^t)^2 \quad (3)$$

where the summation is carried out in all the indices (s, t) corresponding to two adjacent pixels, until the whole image $\mathbf{X}$ is visited.

But what if we start with low-order probabilities taken, for instance, from measurements or from expert knowledge? A Gibbs function is not initially related to low-order probabilities but to general, nonintuitive, local interactions. In this case, we would want to have dependencies amenable to representation by means of a tree. Why? Figure 4 helps justify this idea; to find the joint probability of the variables in the figure, we need only apply the chain rule

$$P(A, B, C, D, E, F) = P(B \mid A, C, D, E, F)\, P(D, E, F \mid C, A)\, P(C \mid A)\, P(A) \quad (4)$$

and, because we are assuming that separation in a graph means conditional independence, we can write

$$P(A, B, C, D, E, F) = P(B \mid A)\, P(D \mid C)\, P(E \mid C)\, P(F \mid C)\, P(C \mid A)\, P(A). \quad (5)$$

This is what we wanted: a joint probability expressed as a product of low-order probabilities, which a human can easily infer from data. Note,

however, this is only possible for tree-structured networks.

But what if things were not so simple—that is, what if graph arrangement is not a tree? We must look for equivalent structures that let us work with low-order probabilities. The key to finding out whether such structures exist is the concept of *chordality*. A graph is said to be chordal if every circle of length four or more nodes has at least one chord—that is, an edge joining two nonconsecutive vertices along that cycle.

If the graph is chordal, the probability model encoded (at least, as an I-map) in the graph is said to be decomposable. Furthermore, we can encode decomposable probability models by an equivalent graph that turns out to be a tree. This tree is called a *join tree* because it is made up of cliques instead of nodes. Pearl posed this as a theorem,[1] which (basically) reads

> If P is decomposable with respect to the graph G, then P can be written as a product of the probabilities of every clique divided by a product of the probabilities of their intersections.

This theorem is also the basis of the *clustering* algorithms[4] that are widely used in DAGs to perform inferences. We will come back to this later.

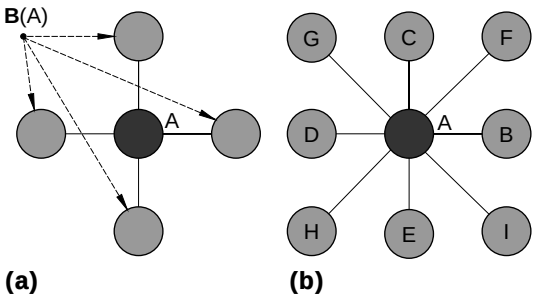

Figure 3. Two possible neighbor configurations for node A.

Selected applications of UGs

Researchers have used UGs (or MRFs) for a myriad of applications. Here are just a few examples to illustrate their applicability.

In the statistical physics community, Nicholas Metropolis and his colleagues pioneered the idea of obtaining sampling distributions in complex systems; they proposed a stochastic method to calculate the properties of any substance consisting of interacting particles.[5] Others have applied their ideas to goals such as visual texture synthesis,[6] where texture is encoded in the spatial relations among pixels. Here, the problem is to generate a sample of the so-defined random field. You can also use algorithms such as Metropolis or the Gibbs sampler.[7]

As an example, Figure 5 shows a sample texture obtained by the Metropolis algorithm. The MRF is defined by means of a neighborhood (shown in Figure 4b) with local energy function

$$H(A) = A\,(-2.0 + 0.05\,(B + C + D + E + F + G + H + I)). \qquad (6)$$

A straightforward application of UGs is as a postprocessor of other segmentation algorithms.[7] If other algorithms give spotty results because of texture or any other reason, the MRF can impose smoothing constraints to filter out the spots. In this case, we can use the Gibbs sampler to estimate the *mode*—the most probable state—that complies with the probability model. We can find the mode by means of the *simulated annealing* algorithm.

Figure 6 illustrates an example of this application using the following local characteristic (given elsewhere[8]):

$$\Pi(A|\mathbf{B}(A)) = \frac{e^{v(A)}}{\sum_{k=1}^{2} e^{v(A_k)}} \qquad (7)$$

where $v(A)$ is the number of neighbors of A in the same state as A. Each state corresponds to a

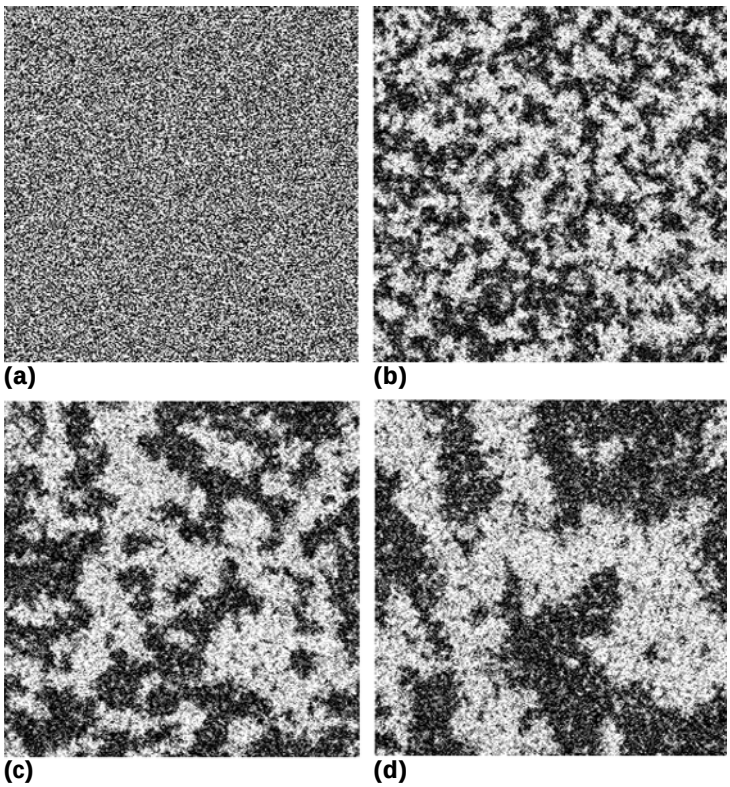

Figure 5. Markovian texture synthesis example with the Metropolis algorithm: (a) the initial configuration, and how it looks after (b) 20, (c) 100, and (d) 500 iterations.

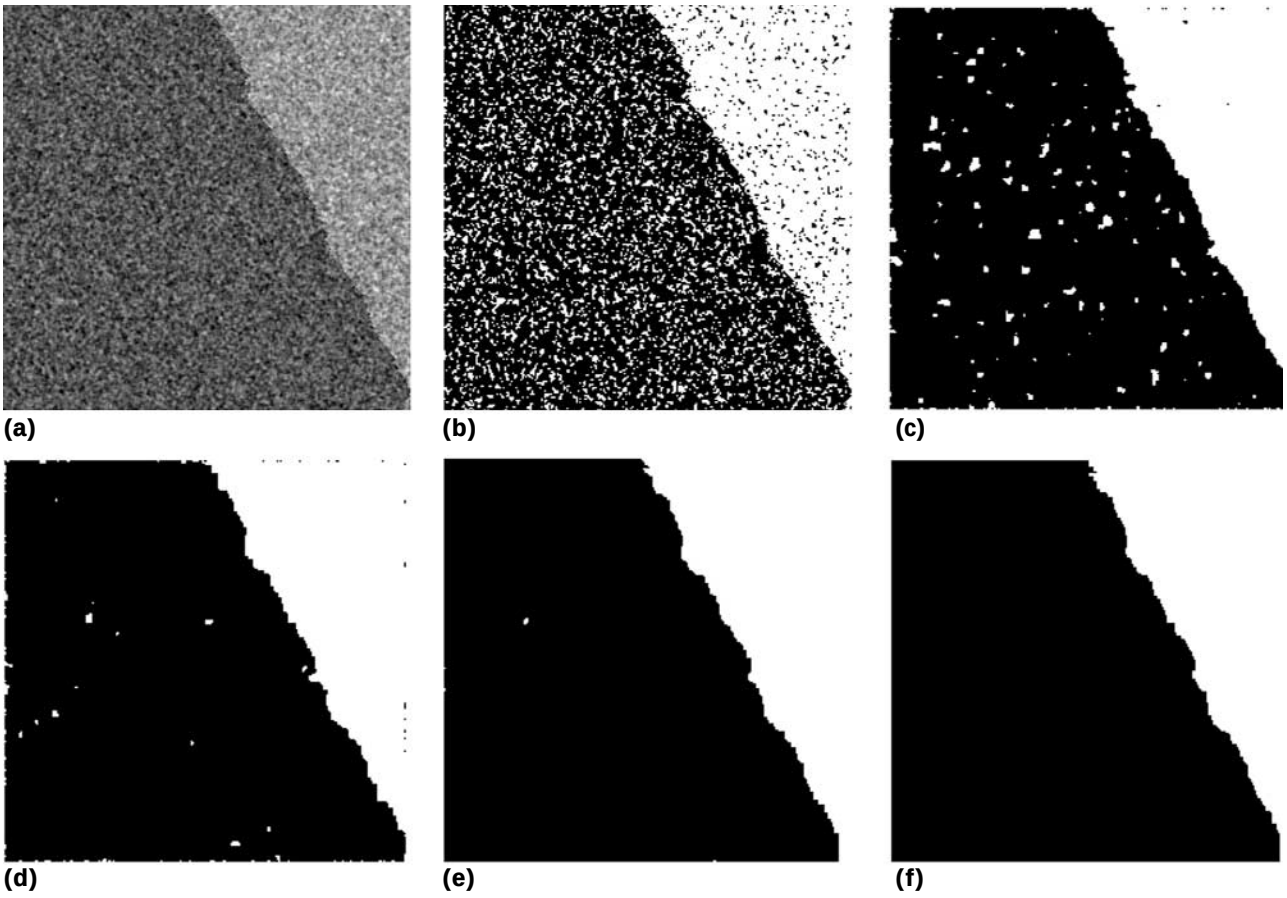

Figure 6. Cleaning a noisy segmentation: the (a) original image, (b) initial segmentation, and cleaning process after (c) 5, (d) 10, (e) 15, and (f) 20 iterations.

region of the segmented image. Thus A can have one of two values, A_k k = {1, 2}. The neighborhood used is Figure 4a's.

A third application is in relaxation processes related to stereo matching. In a stereovision context, we can statistically relate the correspondence of a certain node to the correspondences of nearby nodes (two points in a pair of stereo images correspond or match if they are the projection of the same spatial primitive on each of the images). We can use this relation to describe any correspondence map as an MRF, and we can consider the similarity between matches in the stochastic characterization of the system.[9]

The directed counterpart: DAGs

The counterpart of UGs is a graphical model in which connections are directed. Placing an arrow on an edge brings deep implications, the most important of which is that causality is implied. Arrow-destination nodes (*descendants*) are direct consequences of nodes from which arrows depart (*ascendants*), thereby creating a specific type of dependency, a cause–consequence relation. These networks are also called *causal networks*.

DAGs are based on the concept of d-separation, or d-sep for short. (Eugene Charniak uses the complementary term d-connection.[10]) A group of nodes **Z** is said to d-separate the disjoint groups of nodes **X** and **Y** when either the nodes **Z** are ascendants of both groups **X** and **Y** (see Figure 7a), or **Z** is an intermediate group of nodes (see Figure 7b).

This is not a formal definition but just an intuitive approach to the problem. A DAG is a perfect map of a probability function P if d-separation implies conditional independence and vice versa. In this case, P is said to be DAG-isomorphic. As in the case of UGs, not all probabilistic systems are DAG-isomorphic (see the "Pearl's theoretical framework" sidebar). If only one sense of implication exists—that is, if $d\text{-}sep(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \Rightarrow I_P(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$—then the DAG $B(\mathbf{U}, \vec{\mathbf{E}})$ is said to be an I-map of P.

There are a number of issues to consider in

comparing DAGs and UGs.

Let's go back to the problem of the police speed control schedule depending on the party and the holiday (see Figure 2). We can draw arrows from F and H to SC because both are direct causes of SC. We said that the police control behaved as a cutset between the other two variables. However, according to the definition, SC does not d-separate H from F; in fact, it d-connects them. It is actually the connection that renders them (correctly) as conditionally dependent, even though they are marginally independent.

How do we build a DAG out of a number of variables? We identify which nodes are direct causes of which other nodes. We draw as many nodes **U** as variables in the problem and then draw directed connections $\mathbf{E}$ from every cause to its consequents. There will be a bunch of marginally independent root nodes (from which arrows only depart). Nonroot nodes will be conditionally independent of their nondescendants given an instantiation of their connected ascendants (the parents).

How do we quantify the links? Links are now simply *conditional probability tables* (CPTs): considering node X_i and denoting its parents by Π_{X_i}, all that is needed is $P(X_i | \Pi_{X_i})$. This type of function is clearly tied to empirical data (if available) or to human knowledge.

How can we obtain a probabilistic system from CPTs? If you multiply all CPTs of all descendants by the marginal probabilities of all root nodes, the function you obtain will be a consistent probabilistic system. In our example, $P(SC, F, H) = P(SC | F, H)\, P(F)\, P(H)$. As long as the factors are consistent probabilities, the joint function will also be a consistent probability function.

Not all of these points are advantages for DAGs, though. Recall the notion of chordality, and the fact that chordal probabilistic systems P encoded in chordal graphs must be decomposable. Consider the nonchordal UG depicted in Figure 8. We can infer from the figure that nodes A and C are separated from B and D. If the UG is an I-map of a probabilistic system P, we can conclude that $I_P(A, \{B, D\}, C)$ and $I_P(B, \{A, C\}, D)$. However, try to think of a DAG representing these two dependencies; it is not possible, at least with only these variables.[1] Therefore, we cannot graphically encode all dependencies in DAGs. They might be more intuitive than UGs, but we cannot solve every problem with them.

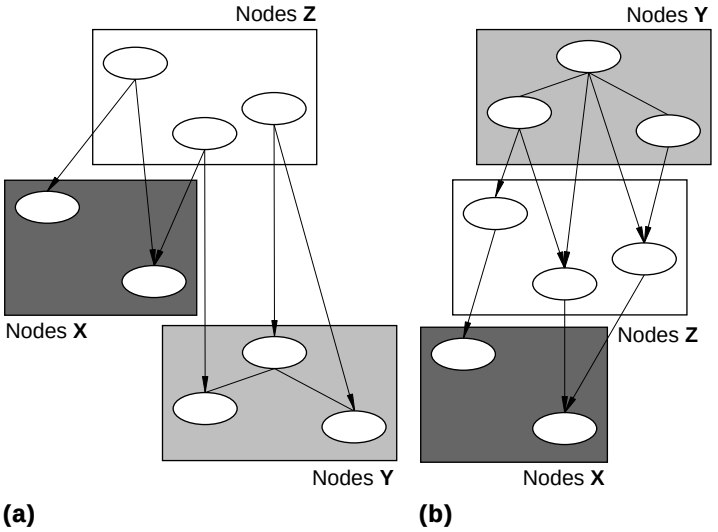

Figure 7. The two cases of d-separation: (a) Z nodes are ascendants of X and Y, or (b) Z is an intermediate group of nodes.

Inference in DAGs

The straightforward application of DAGs is to encode the available knowledge and then to try to assess other probabilities not directly available from the knowledge itself. As we mentioned earlier, human knowledge is basically local; therefore we must use techniques to obtain consequences beyond those directly available in the models' CPTs. Inference in DAGs is carried out by means of the Bayes rule and other well-known operations of basic probability theory (as a consequence, these networks are also known as *Bayesian networks*).

The process is not conceptually complicated. Let's come back to our example and make things a bit harder: we will add a node GT representing the fact that I might get a ticket (for speeding or for any other reason). In our model, I need to specify the CPT of GT conditioned only on SC, since SC is the only direct cause of my getting a ticket (see Figure 9).

Let's assume that I got a ticket, that my friend Jane knows this, that Jane wants to know whether the ticket was for speeding, but that Jane does not want to ask me directly. She would want to know how likely it is that a speed control was on the road, now that she knows I got a ticket. Straightforward application of the Bayes

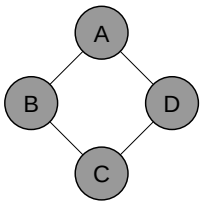

Figure 8. A nonchordal graph without a DAG counterpart.

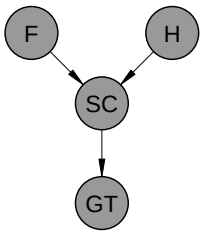

Figure 9. This DAG includes nodes for fiesta (F), holiday (H), speed control (SC), and GotTicket (GT).

rule (for binary variables using the convention $P(SC|\underline{GT}=1) = P(SC|gt)$ and $P(SC|GT=0) = P(SC|\overline{gt})$) gives

$$P(SC|gt) = \frac{P(gt|SC)P(SC)}{P(gt)} \tag{8}$$

$$\propto (gt|SC)P(SC) \tag{9}$$

Using Equation 9, we can avoid the calculation of $P(gt)$, because we can obtain the proportionality factor by forcing

$$P(sc|gt) + P(\overline{sc}|gt) = 1 .$$

We can now complete these calculations by taking into account the other two nodes in the network:

$$P(SC|gt) \propto P(gt|SC) \times \sum_{F,H} P(SC|F,H)P(F)P(H) \tag{10}$$

where we have assumed independence between the root nodes. Notice that the inference process performs a summation in all the CPT entries of node SC. It is then obvious that inference complexity will be related to the number of conditions of a node (the parents of the nodes involved in the inference). This procedure is simply a node-elimination task; the idea is to sum out the contributions of all those nodes not present in the probability to be calculated. The order in which these nodes are eliminated is important, because the complexity associated with the algorithm is a function of the number of entries in the CPTs. The more arrows converge in a node, the larger the number of conditionants in the CPT of that node and, consequently, the larger the inherent complexity. We might therefore think of criteria to efficiently eliminate nodes so as to minimize the average number of conditionants in the whole procedure.

However, inference in a general case (in a *multiply connected network*, with no restrictions—that is, with arbitrary topologies) is an NP-complete problem.[11] So what do we do?

Dealing with complexity

One of the most popular algorithms is Pearl's message-passing scheme.[12] The major breakthrough of his algorithm was to change the concept of a network from the traditional, passive scheme that encodes only data to a novel, active, information-flowing scenario in which gathered evidence triggers an updating procedure over the whole network. In addition, the message-passing algorithm is strictly local: a node absorbs the effect of evidence by means of a number of messages from father and siblings and then floods its updated information back to its father and siblings.

Convergence to a stable solution is guaranteed when the network is singly connected (there is no more than one path between two nodes in the underlying undirected graph). Why? The reason is buried in the algorithm details. However, we can give you a hint: in our example, suppose that parties are more often held when the day after is a holiday than on a regular day. We model this by adding an arrow originating at H and pointing at F. Suppose that we find out that a speed control is deployed on the road. In this case, messages emanating from SC are propagated up and down. We will have a loop in which messages from SC go through F, arrive at H, and keep flowing down to SC—and the process continues with no end.

As you can see, the presence of loops makes the message-passing scheme unstable. But, even if the system were stable, we could not trust the results of such an inference process. A typical example is the following: I tell A that I suspect something; A tells B; then B tells me about it. Should I increase my suspicions? If I don't know that B knows it because of me, I would! But this is just an illusion of evidence: I am just hearing the message that I sent.

Handling loops

In cases where loops are part of the DAG, we must use a different inference scheme. A popular alternative is known as *clustering*. Algorithms based on this philosophy make use of the fact that we can structure any DAG as a tree of clusters as long as we do not limit the cluster size.

Actually, simply by clustering all but the leaf nodes in a DAG, we end up with a tree structure.

Steffen L. Lauritzen and David J. Spiegelhalter describe an efficient clustering method whose purpose is to build a join tree from a chordal graph.[4] As we said earlier, a chordal graph represents a decomposable probability model. Therefore, we can base an efficient inference procedure on finding an equivalent graphical model that is chordal. Once you do this, you end up with a join tree (an undirected graph) that is free of loops; then you can use Pearl's local message-passing scheme in a clique space (as opposed to a node space). The price you pay is the larger number of edges needed to make the graph chordal.

This algorithm has been generalized both by Finn V. Jensen and his colleagues[13] and by A. Philip Dawid.[14] The Viterbi algorithm and the forward–backward algorithm, both widely used in the signal-processing community, are particular cases of these algorithms.[15]

Another alternative is based on obtaining approximate inferences—that is, approximate probabilities given instantiated nodes. These algorithms are based on stochastic simulation, an example of which is the Gibbs sampler. Even though approximate inference is also an NP-hard problem,[16] an important effort has focused on finding approximate algorithms that perform efficiently in DAGs and give reliable results faster than exact algorithms in large networks.

The main idea of these algorithms is to artificially create network instances that let us approximate probabilities of interest by averaging results. In *logic sampling*,[17] for example, you start at the root nodes and create data according to the nodes' probabilities. You then sequentially visit the rest of the nodes and create instances according to the instances of the parents and to the nodes' CPTs. The process continues until you achieve stable-enough values of the queries.

Its straightforwardness makes it suffer a number of drawbacks; the main one is that you might obtain instances that do not comply with evidence. Consider this scenario in our speed control example: having received a ticket, you want to obtain an approximate probability of tomorrow being a holiday. The approach based on stochastic simulation starts creating instances at the root nodes H and F and goes downward, creating instances according to the values of parents and the CPTs. However, this approach does not force the node GT to comply with the observation. Therefore, you should discard trials that don't comply with evidence.

The likelihood weighting algorithm[18] avoids discarding trials that do not comply with evidence. It is based on the same idea, but it weighs network instances that are in conflict with the evidence according to the CPTs' values. This algorithm leads to unbiased estimates of the desired probabilities.

A lot of variants on these two algorithms exist that create data in different ways to speed up the convergence process.[19] Of main concern is how to estimate the number of trials that are needed to obtain convergence to the real probabilities.[20] This is a current research issue. You can find an excellent tutorial on stochastic simulation in our reference list.[21]

Applications

A variety of applications from different communities demonstrate DAGs' potential usefulness.

Software debugging

When we encounter a problem during a complex program's execution, we debug the software. However, complex systems have many alternative paths that we must explore; a full, sequential search is not desirable. If software-debugging experts were to encode their experience into a DAG, we could weigh alternatives by the likelihood that they contain the error, provided we observe a symptom of the error. This is the main philosophy of the DAAC (dump analysis and consulting) system.[22]

We can extend this idea to other troubleshooting problems. To make a more realistic model, we can assign costs to different actions, and then, after seeing the evidence, repair the system using a minimum-average-cost policy.[23]

Using a DAG for such a problem is just an alternative to using a decision tree. Even though decision trees are probably more intuitive, they have a huge number of branches in problems with numerous variables. A DAG is a more parsimonious structure to encode these dependencies. These special DAGs are called *influence diagrams*.

Information retrieval

The task of information retrieval[24] is to quickly obtain the pieces of information that are relevant to a certain topic, along with a measure of relevance so that the user can focus on the most interesting documents. We can encode this with a Bayesian network in, for instance, a library, where documents will be (or not be) relevant to a certain topic. If a document is relevant to a given

topic, it likely contains certain features that the user will introduce as elements of the query. Therefore, we can now pose the problem as finding the probability of a document's relevance to a topic, provided it contains some of the features that the user has input. We use the measure of probability as a rank of document relevance.

Medical applications

A number of health care applications use DAGs.[4] For example, R. Riccardo Bellazi[25] used a Bayesian network with continuously valued nodes to propose an optimal schedule of r-HuEPO drug delivery. (Now CPTs are replaced with conditional density functions.) r-HuEPO can take over the kidney's task of synthesizing erythropoietin, a hormone concerned with red blood cell production. Deficiency of this hormone is believed to be a major cause of anemia and can result from kidney malfunction.

Bellazi has derived a stochastic model that relates the human body's sensitivity to the EPO with the hemoglobin concentration in the blood. This sensitivity is a function of several parameters. The network's task is to infer the model parameters from a population and to probabilistically adapt them to a specific patient (taking into account the person's history). Then the network predicts the patient's response to r-HuEPO. A comparison of predictions and observations results in an optimal policy of drug delivery.

Information fusion

Manfred Prantl, Harald Ganster, and Axel Pinz use a DAG to decide which pieces of information to use for a multispectral terrain classification problem.[26] Using at most five uncorrelated sensors, the authors obtained measurements of several terrains. The problem was to classify the sample terrains into one of the prespecified classes after observing the sensor values. This approach tries to quantify the number of sensors to use—that is, can only one or two sensors collect enough evidence to make a reliable decision? This concept is called *active fusion*—fusion as a function of the certainty of information. The network encodes the probabilities of observing a pair of values of mean and variance from a sensor, given that a specific terrain is encountered. We used the network to minimize the number of observations needed to classify with a given confidence. This is done by keeping track of the posterior probability of a class given the evidence. When this probability exceeds a threshold, no more sensors are needed.

Although you can probably solve some of the problems just presented using other methods, DAGs work well in a lot of problems mainly because of their ease at incorporating human knowledge in a probabilistic framework. The problem of optimality of these structures is a matter for discussion.

What is the future of graphical models? This is hard to answer. However, the fact that we can pose well-known algorithms in other fields as particular cases of general-purpose algorithms in graphical modeling makes these computational structures an interesting theoretical tool, and a powerful practical tool as well. So we can expect progress in different tiers: theoretical results that demonstrate coincidences between classical results and algorithms on graphical models, more efficient exact and approximated inference algorithms, and above all, models of complex dynamical systems that can learn probabilities from data and automatically reconfigure their structures as new evidence comes in. The Web sites listed in the accompanying sidebar "For further information" offer excellent entry points for papers, software, and freeware. CiSE

Acknowledgments

The Spanish CICYT under research grants TIC97-0772 and 1FD98-0881 and Junta de Castilla y León under research grant VA 78/99 partially funded this work.

References

1. J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Mateo, Calif., 1988.
2. J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems," *J. Royal Statistical Soc., Series B*, Vol. 34, 1974, pp. 192–236.
3. G. Winkler, *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, Springer-Verlag, New York, 1995.
4. S. Lauritzen and D. Spiegelhalter, "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems," *J. Royal Statistical Soc., Series B*, Vol. 50, No. 2, 1998, pp. 157–224.
5. N. Metropolis et al., "Equation of State Calculations by Fast Computing Machines," *J. Chemical Physics*, Vol. 21, No. 6, June 1953, pp. 1087–1092.

6. G. Cross and A.K. Jain, "Markov Random Fields Texture Models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 1, Jan. 1983, pp. 39–55.
7. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 6, Nov. 1984, pp. 721–741.
8. J. Zhang, J.W. Modestino, and D.A. Langan, "Maximum-Likelihood Parameter Estimation for Unsupervised Stochastic Model-Based Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 3, No. 4, July 1994, pp. 404–419.
9. L. Tardón, J. Portillo, and C. Alberola, "Markov Random Fields and the Disparity Gradient Applied to Stereo Correspondence," *Proc. IEEE Int'l Conf. on Image Processing, Vol. III*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1999, pp. 901–905.
10. E. Charniak, "Bayesian Networks without Tears," *AI Magazine*, Winter 1991, pp. 50–63.
11. G.F. Cooper, "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence*, Vol. 42, Nos. 2–3, Mar. 1990, pp. 393–405.
12. J. Pearl, "Fusion, Propagation and Structuring in Belief Networks," *Artificial Intelligence*, Vol. 29, No. 3, Sept. 1986, pp. 241–288.
13. F. Jensen, S. Lauritzen, and K. Olesen, "Bayesian Updating in Recursive Graphical Models by Local Computations," *Computational Statistical Quarterly*, Vol. 4, 1990, pp. 269–282.
14. A. Dawid, "Applications of a General Propagation Algorithm for Probabilistic Expert Systems," *Statistics and Computing*, Vol. 2, 1992, pp. 25–36.
15. P. Smyth, D. Heckerman, and M.I. Jordan, "Probabilistic Independence Networks for Hidden Markov Probability Models," *Neural Computation*, Vol. 9, No. 2, 1997, pp. 227–269.
16. P. Dagum and M. Luby, "Approximating Probabilistic Inference in Bayesian Belief Networks is NP-Hard," *Artificial Intelligence*, Vol. 60, No. 1, Mar. 1993, pp. 141–153.
17. M. Henrion, "Propagation of Uncertainty in Bayesian Networks by Probabilistic Logic Sampling," *Uncertainty in Artificial Intelligence*, Vol. 2, 1988, pp. 149–163.
18. R. Fung and K. Chang, "Weighting an Integrating Evidence for Stochastic Simulation in Bayesian Networks," *Proc. Fifth Conf. Uncertainty in Artificial Intelligence*, AAAI Press, Menlo Park, Calif. 1988.
19. L. Hernandez, S. Moral, and A. Salmeron, "Mixing Exact and Importance Sampling Propagation Algorithms in Dependence Graphs," *Int'l J. Intelligent Systems*, Vol. 12, 1997, pp. 629–653.
20. M. Pradham and P. Dagum, "Optimal Monte Carlo Estimation of Belief Network Inference," *Proc. 12th Conf. Uncertainty in AI* (UAI '96), Morgan Kaufmann, San Francisco, 1996, pp. 446–453.
21. S. Cousins, W. Chen, and M. Frisse, "A Tutorial Introduction to Stochastic Simulation Algorithms for Belief Networks," *Artificial Intelligence in Medicine*, Vol. 5, 1993, pp. 315–340.
22. L. Burnell and E. Horvitz, "Structure and Chance: Melding Logic and Probability for Software Debugging," *Comm. ACM*, Vol. 38, No. 3, Mar. 1995, pp. 31–41.
23. D. Heckerman, J. Breese, and K. Rommelse, "Decision-Theoretic Troubleshooting," *Comm. ACM*, Vol. 38, No. 3, Mar. 1995, pp. 49–57.
24. R. Fung and B. del Favero, "Applying Bayesian Networks to Information Retrieval," *Comm. ACM*, Vol. 38, No. 3, Mar. 1995, pp. 42–48.
25. R. Bellazi, "Drug Delivery Optimization through Bayesian Networks," *Computers and Biomedical Research*, Vol. 26, 1993, pp. 274–293.
26. M. Prantl, H. Ganster, and A. Pinz, "Active Fusion Using Bayesian Networks Applied to Multi-Temporal Remote Sensing Imagery," *Int'l Conf. Pattern Recognition* (ICPR '96), Vol. C, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 890–894.

For further information

The following links are current as of May 2000:

A Guide to the Literature of Probabilistic Methods for Decision Support Systems: www.cs.pitt.edu/~tsamard/bnpointers.html
Bayesian Networks and Decision-Theoretic Reasoning for Artificial Intelligence: robotics.stanford.edu/~koller/BNtut
Microsoft Belief Network Tools: www.research.microsoft.com/research/dtg/msbn/default.htm
Software for Manipulating Belief Networks: bayes.stat.washington.edu/almond/belief.html
Association for Uncertainty in Artificial Intelligence: www.auai.org
AUAI Tutorials and Survey Sources: www.auai.org/auai-tutes.html
Bayesian Knowledge Discovery Project: kmi.open.ac.uk/projects/bkd
Publications List of the Research Group of Uncertainty Treatment in Artificial Intelligence, Univ. of Granada: decsai.ugr.es/gte/publication.html

Carlos Alberola is an associate professor at the ETSI Telecomunicacion of the University of Valladolid, Spain. He has coauthored other publications in the IEEE and IEE societies, and his research interests include statistical signal and image processing applications in medicine. He received his BS and PhD in telecommunications engineering from the Polytechnic University of Madrid. Contact him at ETSI Telecomunicacion, Campus Miguel Delibes s/n Univ. de Valladolid, Valladolid, 47011 Spain; carlos@tel.uva.es.

Lorenzo Tardón works in the Department of Communications Engineering at the University of Malaga. His research interests include image processing, stereoscopic imaging, Markov random fields, wavelets, and spread spectrum communications. He received his MSc and PhD in telecommunications engineering from the University of Valladolid and the Polytechnic University of Madrid, respectively. Contact him at lorenzo@ic.uma.es.

Juan Ruiz-Alzola is an associate professor in the Department of Signals and Communications in the University of Las Palmas de Gran Canaria. His research interests include signal processing, statistical methods, physical modeling, computer vision, computer graphics, and their application to medical imaging. He received his BS and PhD degrees in telecommunications engineering from the Polytechnic University of Madrid. Contact him at jruiz@dsc.ulpgc.es.