Research Report S-121

Fusion and Propagation in Graphical Belief Models

Russell Almond

Harvard University
Department of Statistics

June, 1988

This work was facilitated in part by Army Research Contract DAAL03-86K-0042. Portions of this paper were presented at the 1988 Interface Conference in Reston, VA.

ABSTRACT

Graphical models give a clear and concise way of describing dependencies among many variables. Only relationships among variables which all share a common hyperedge must be modeled, considerably simplifying both the modeling and the computational task. Graphical models have been studied by Pearl [1986a,1986b], Moussouris[1974], and Lauritzen and Spiegelhalter[1988] in the Bayesian case, and Kong[1986a], Shafer, Shenoy, and Mellouli [1986] and Shenoy and Shafer[1986] in the Belief Function case.

Belief functions are a generalization of probability that allow ways to express total ignorance, Bayesian prior probability distributions, conditional probability distributions (likelihoods), logical relationships (production rules) and observations. All these diverse types of knowledge can be combined with a uniform fusion rule, the direct sum operator. Belief functions can be simply restricted to a smaller frame and easily extended to a larger frame without adding additional information. The theory of belief functions is developed in Shafer[1976,1982] and Kong[1986a].

By a simple procedure given here and in Kong[1986b], we can transform the model hypergraph into a *tree of closures*. This is a tree of “chunks” of the original problem, each “chunk” can be computed independently of all other chunks except its neighbors. Each node in the tree of closures passes messages (expressed as belief functions) to its neighbors consisting of the local information fused with all the information that has propagated through the other branches of the tree. Using this propagation algorithm along with the fusion algorithm given by the direct sum operator, we can easily compute marginal beliefs, and can save considerable computational cost over the brute force approach.

Key Concepts: *Graphical Models, Belief Functions, Bayesian Models, Fusion and Propagation, Probability in Expert Systems, Triangulated Graphs.*

Fusion and Propagation in Graphical Belief Models

Russell Almond
Harvard University
Department of Statistics

1. Attacking Large Problems

We are looking for ways to attack a class of large problems that comes up frequently in expert systems; that is, we are looking at problems with a large number of variables, but few repetitions of those variables, something that conventional statistical methodology is not well equipped to handle. For example suppose we have a problem with N attributes or variables, and M independent relationships among groups of those attributes. Suppose we further let Θ be the joint outcome space for those N variables, and we model the relationships as a belief function over Θ. Then to compute the combined belief function, $\mathrm{BEL}_{\mathcal{G}}$, from which we could get marginal belief functions for any attribute or group of attributes of interest, the computational cost is $M \cdot 2^{|\Theta|}$. For the special case where all of the attributes are binary variables, we have the cost $M \cdot 2^{2^N}$.

Since these problems are too large to be easily computed, we seek to apply the following strategem to make the computational tasks manageable:

1. Break up the problem using the *Graphical Modeling* and conditional independence assumptions, as described in Section 2.
2. Locally model relationships with *Belief Functions*. This process will be briefly described in Section 3.
3. Re-express the graphical model as a *Tree of Cliques* (see Dempster and Kong[1988] and Kong[1986b]). The tree of cliques will be described in Section 4.
4. *Propagate* and *Fuse* local information to find margins of the total belief function. This will be described in sections 5 and 6.

Let m be the number of nodes in the Tree of Cliques ($m > M$ but only slightly), and k be the maximum number of neighbors of a chunk in the tree. Furthermore, let C^* be the largest chunk in the tree, n be the number of variables in C^*, and Θ_{C^*} be the outcome space associated with C^*. Then our computational costs are $\leq m \cdot k \cdot 2^{|\Theta_{C^*}|}$ or for the case of binary variables, $m \cdot k \cdot 2^{2^n}$. In most cases, $n \ll N$, and we are reducing the size of the double exponential, therefore using our method yields big time savings. Furthermore, these computational costs are worst case figures, based on complex belief functions. In practice, with simple belief functions, the computational costs will be much smaller.

5. We are now finished, and can look at the margins of our total belief function. Section 7 shows one margin for a simple example.

We will follow a simple example throughout the course of this paper. Let us consider the reasoning by which the Captain of a ship decides how many days late her ship will arrive in port. First let us define the attributes, or variables of the problem. Our goal is to find the *Arrival delay*, or by how many days the ship will be delayed (For simplicity we assume it will be an integer). This delay is the sum of two pieces, the *Departure delay* and the *Sailing delay*. Before the ship leaves port it could be delayed for *Loading* problems, a *Forecast* of foul weather could cause the Captain to delay departure, and *Maintenance* could cause the ship to sit at the dock. For simplicity we will assume that each of these three factors delay departure by one day. Therefore the total *Departure delay* could be up to three days. Similarly, bad *Weather* en route could cause delays, as could needing to make *Repairs* at sea. These delays contribute to the *Sailing* delays, again an integer number of days. We will develop example as the paper proceeds. Appendix III will give the details of the model.

2. Graphical Models

Graphical models provide a way of organizing information about the relationships among variables in problem domains with many variables. Thus we are thinking about applications with few repetitions of a large number of variables. Decision problems, diagnosis problems, fault trees, log-linear models, and expert systems all fall into this category. We could say that the graphical model is a form of knowledge representation, and notice that relational data base design very much resembles graphical model design.

When we design a graphical model, we break a large problem (the complete model) up into a series of smaller problems. We then organize the small problems using a hypergraph. Each node of the model hypergraph is an attribute or variable of the problem. Each edge of the model hypergraph corresponds to a group of attributes that are related through some mechanism, we will later model those mechanism with the methods of Section 3. Any pair of attributes (nodes) which are not directly connected are assumed to be conditionally independent under the Markov conditions (see below). This last part is important. It means that we only need to define belief function mechanisms by which attributes that share a common hyperedge are connected; we will be able to derive the mechanism by which the attributes which do not share a common hyperedge are connected from the information found in the hyperedges. Thus we can concentrate on the small problems. Furthermore, Pearl[1982] claims that independence conditions are often easier to elicit from an expert than joint distributions. Thus building a graphical model limits the size of the joint distributions (or complex mechanisms) which we must elicit.

For example, figure (1) shows the model hypergraph for the Captain's decision, with the first letter of each attribute name representing the node.

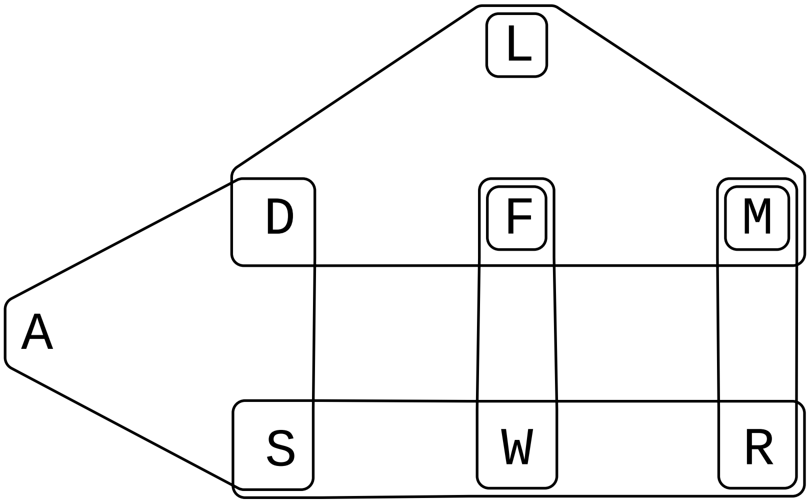


Figure 1. Model Hypergraph for Captain's Decision

Now let us examine the edges, beginning with the edge containing, *L*, *D*, *F* and *M*. In the previous paragraph we noted that *L*_oading_, weather *F*_orecast_, and *M*_aintenance_ delays could each add one day to our total *D*_eparture delay_. This edge represents the expression of that rule. Similarly, the edge containing *S*, *W* and *R* expresses of a similar rule for delays that occur at sea. The edge containing *A*, *D* and *S* represents the logical assumption that the total delay equals the sum of the two partial delays. The edge containing *F* and *W* represents the condition that the *Weather* and the *Forecast* are dependent variables, and similarly the edge between *M* and *R* graphically depicts the dependence between *Maintenance* and *Repairs* at sea. Lastly the three singleton edges containing *L*, *F* and *M* respectively represent our prior beliefs about *Loading*, weather *Forecast*, and *Maintenance*.

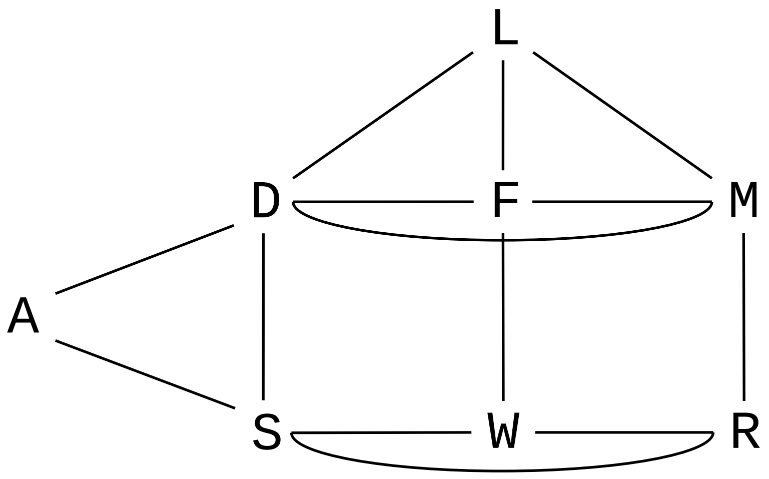


Figure 2. 2-Section of Model Hypergraph for Captain's Decision

Often it is convenient to show a two-section of a hypergraph; that is a simple graph which has the same connectivity as the hypergraph. The two-section for the Captain's decision example is shown in figure (2).

It is customary to assume that the model hypergraph is Markov. Moussouris[1974] develops the conditions under which a graph is Markov. The definition extended to hypergraphs follows.

Markov Conditions. *Let $\mathcal{G}$ be a hypergraph. For any pair of nodes X and Y of $\mathcal{G}$, such that X and Y do not share a common hyperedge, we let S be a set of nodes such that all paths from X to Y pass through S. For any such pair,* $\mathrm{BEL}(X \mid S, Y) = \mathrm{BEL}(X \mid S)$ *and* $\mathrm{BEL}(Y \mid S, X) = \mathrm{BEL}(Y \mid S)$. *That is X is independent of Y given S. Then $\mathcal{G}$ will be considered a* Markov Hypergraph *and is said to satisfy the* Markov Conditions*.*

Let us illustrate this with an example. Consider the two attributes R and D. We can see that the set of attributes, $\{S, F, M\}$ separate the two target attributes. Thus if we hold S, F, and M fixed, R and D are independent. This makes sense in terms of the original model, since, if we know the *S̲ailing delay*, the *F̲orecast* for the weather, and the *M̲aintenance* record at the dock, it is plausible that the *D̲eparture delay* and the *R̲epairs at sea* are independent. In fact we could make stronger independence statements, but that would rely on specific knowledge about the dependences represented within the edges, and cannot be inferred from the graph.

These Markov Conditions allows us to model the interactions in one piece of the problem at a time. The whole model can be assembled later as a graphical model. For the rest of this paper, we will assume that the model hypergraph, $\mathcal{G}$ is Markov.

3. Belief Functions

Here I will briefly discuss why belief functions are useful in graphical models. Definitions and detailed descriptions can be found in Shafer[1976] or Kong[1986a]. I will only give an abstract of the ideas here. Appendix I will describe belief functions in sufficient detail to follow the example.

1. Upper and Lower Probabilities. A belief function provides a two point estimate of an unknown probability, the belief (BEL) or lower probability and the plausibility (PL) or upper probability, not simply a single number. This allows us to express uncertainty about the chance of an event occurring in simple way.

2. Can be used to represent: Bayesian Probabilities (we call them Bayesian because they are probabilities being used in the subjective or Bayesian manner.), *Logical rules with or without "confidence factors"* (Confidence Factors are probabilities that given rules hold.), *Observations, and Ignorance* (A "Vacuous" belief function provides an unambiguous definition of ignorance, unlike a so-called non-informative prior). The important feature we are here using is that belief functions incorporate both set and probability theory, hence we can easily mix logical and probabilistic knowledge in a single uniform framework.

3. We can easily Marginalize *to smaller frames or Vacuously Extend to large frames.* Note that the former is true of probabilities as well as belief functions, but the latter is not.

4. $\oplus$ *Fusion Rule.* The direct sum operator involves both set intersection and multiplication of probabilities with renormalization. Hence it is a generalization of both logical and Bayesian inference rules.

Belief functions generally give us a great deal of flexibility at the cost of more complexity in both notation and computation. A general belief function is often a difficult thing to describe or interpret, simply because of the large numbers of sets of outcomes we could examine. On the other hand, we can easily interpret them in some special cases. All the examples in (2) above are easy to specify. Furthermore for a binary variable, the belief function is again easy to interpret; it is simply an upper and lower probability. The binary variable case seems to be, in general, a case were the Bayesian description gives us too few parameters (we must choose a single probability) or else too many (we must specify a prior distribution for all possible values of the probability that the value will be true).

Of course if we restrict our input belief functions to Bayesian probabilities, then we will get probabilities out of the computation. Thus in a general way, every thing we discuss here applies to probabilities as well as belief functions. Doing the mathematics in the belief function notation, helps us to understand what is happening without worrying about difficult technical details about extending probability distributions.

Appendix III shows belief functions constructed for each edge of the Captain's Decision problem.

4. Make *Tree of Cliques*

For our model hypergraph, $\mathcal{G}$, let us choose a collection of sets of the attributes of $\mathcal{G}$, $\mathcal{A} = \{A_1, \ldots A_n\}$, $\mathcal{C} = \{C^1, \ldots C^m\}$. If our model hypergraph is triangulated (acyclic), then the sets C^s will be the cliques (maximally complete subgraphs) of the model hypergraph. If our model is not triangulated, we will follow the procedure given in Kong[1986b] to produce these sets. (This procedure is described in Appendix II). The Kong procedure implicitly fills in hyperedges to create a triangulated graph; the C^s's are cliques of the triangulated graph. The Kong procedure also connects the cliques into a tree. The connections are done in such a was as to satisfy a separation property which will be given below. This tree is called the *Tree of Cliques* and it turns out that for computational purposes, the Tree of Cliques is easier to use than the model hypergraph (This is also described in Dempster and Kong[1988]).

A useful way to think about the tree of cliques, is to consider each clique to be a group of attributes within which some complex interaction takes place. This complex interaction will be modeled by a belief function representing the information local to that clique and by messages, also in the form of belief functions, passed from the neighboring cliques in the tree. Calculations are performed by propagating messages between the cliques via the schema given in Section 5 and by fusing the messages via the schema given in Section 6. The result is a belief function representing the margin of the graphical belief function for each of the margins C^s.

In order to make the computations more modular, we *augment* the tree of cliques by adding each of the original edges of the model hypergraph to $\mathcal{C}$, and connect each of them to any clique that contains it. (Note that every hyperedge will always be contained in at least one clique). We can also augment the tree of cliques by adding any set of attributes that is a subset of one of the cliques. In particular, we can add each of the singleton sets representing one attribute. These singleton sets represents margins of the graphical belief function that might be important to examine later on. We will call the nodes of the augmented tree of cliques, $\mathcal{C}^+$

Each set, $C^s \in \mathcal{C}^+$ has a local belief function, BEL_{C^s}, representing the local information attached. This local belief function can be easily found from our graphical model, if we require that every edge of the original model hypergraph be added to $\mathcal{C}^+$. For every node, C^s, in the augmented tree of cliques if that node corresponds to one of the original edges, then BEL_{C^s} is the belief function corresponding to that edge. If that node is not one of the original hyperedges, then BEL_{C^s} is vacuous.

As we noticed above, the tree of cliques is easy to build if the model hypergraph is triangulated. In order to get the tree of cliques from a non-triangulated graph, we must fill in extra hyperedges to make a triangulated graph. There is often more than one way to fill in the hypergraph and different fill-ins lead to different trees of cliques, some of which are better than others. Because, as we discussed in the first section, the cost of combining belief functions is exponential with the size of the largest clique, trees with smaller cliques will be better. The problem of finding the optimal tree of cliques is NP-hard. Kong and I have developed some heuristics for finding good Trees of Cliques that seem to do well. (Given in Appendix II).

Let us return to the Captain's Decision problem. Figure (3) reproduces the model hypergraph from figure (1). Notice that some new edges (the dashed lines) have been implicitly filled in as part of the construction procedure. Figure (4) shows the augmented tree of cliques.

The nodes $\{A, S, D\}$, $\{S, D, M, F\}$, $\{L, D, M, F\}$, $\{R, S, M, W\}$, and $\{S, M, W, F\}$ are the cliques of the graph in figure (3). The nodes, $\{A, S, D\}$, $\{L, D, M, F\}$, $\{R, M\}$, $\{S, W, R\}$, and $\{W, F\}$ all correspond to original edges of the model hypergraph. They get loaded with the belief functions corresponding to those edges. Furthermore, $\{M\}$, $\{L\}$, and $\{F\}$ all have univariate prior belief functions associated with them, which are also loaded into the appropriate cliques. The edges $\{S, D, M, F\}$, $\{S, M, W, F\}$ and $\{R, S, M, W\}$ are all associated with the filled in edges. They have vacuous belief functions associated with them. The remaining nodes, corresponding to the remaining singleton edges, also have vacuous belief functions associted.

As we mentioned above, the nodes, C^*, of the tree of cliques are connected in such a way as to satisfy the separation property. This is also true of the augmented tree of cliques. Let us define the Separation Property here.

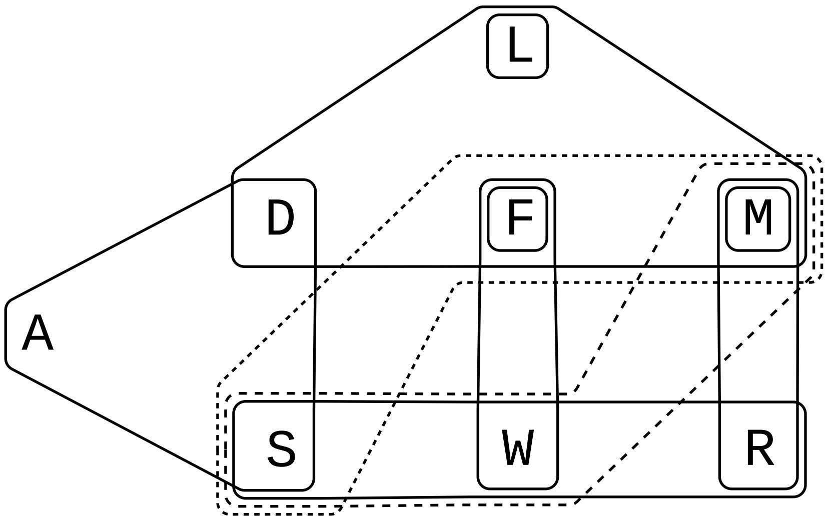


Figure 3. Model Hypergraph with fill-ins.

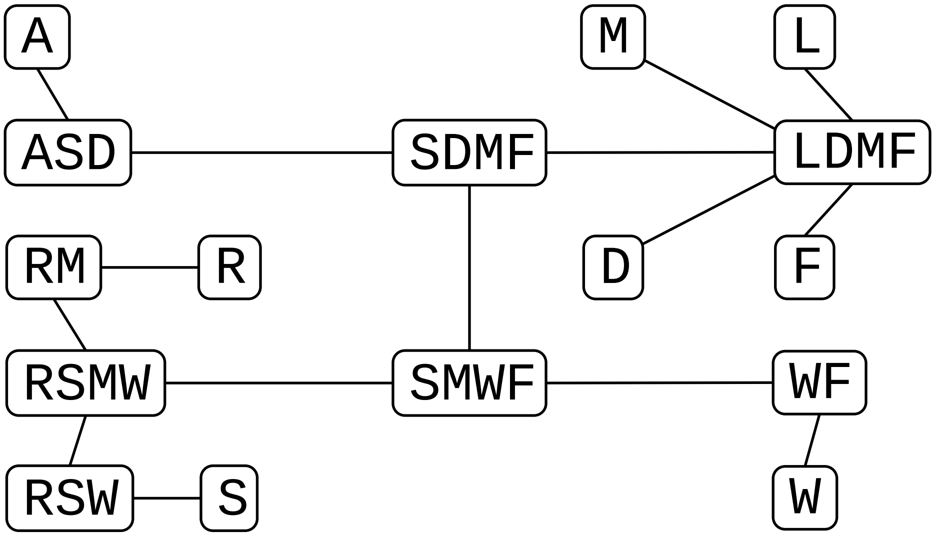


Figure 4. Augmented Tree of Cliques

Separation Property. *Given two nodes in $\mathcal{C}^+$:*

$$C^1 = \{A_{i_1}, \ldots, A_{i_n}\}$$
$$C^2 = \{A_{j_1}, \ldots, A_{j_m}\}$$

Let C^* be any clique lying on the path between them. Then:

$$C^1 \cap C^2 \subseteq C^*$$

In other words, all the cliques in a path between any two cliques contains their intersection. The separation property also implies that if we look at the subgraph of the tree of cliques consisting of all cliques that contain a given attribute, or set of attributes it will be connected. This is not obvious, but can be seen after a few minutes of observation. The fact that the tree of cliques has the separation property can by seen from Kong[1986b].

5. Propagation

We now get to the heart of the computational system, the messages that are passed between the cliques (nodes) of the tree of cliques. Dempster and Kong[1988] describe a system for passing messages between cliques to propagate the local information (which makes up the graphical model) to global information which can be used to answer questions about the process being modeled. Their system operates as follows.

The Cliques pass "messages" to their neighbors in the Tree of Cliques, in the form of belief functions. We will call the belief function passed from C^i to C^j, $\mathrm{BEL}_{C^i \Rightarrow C^j}$, and it will be over the frame corresponding to $C^i \cap C^j$. Each clique "fuses" its incoming messages with its local information, and "propagates" the results as its outgoing message. The fusion step will be described in detail in the next section.

We can calculate the message $\mathrm{BEL} C^i \Rightarrow C^j$, when the node C^i has received messages from all its neighbors except C^j. Therefore the outermost leaves of the tree can immediately pass their messages inwards. The outermost cliques (the leaves of the tree) pass their information toward the center (root). When all the information reaches the center, the cliques in the center start passing messages back towards the outside (leaves).

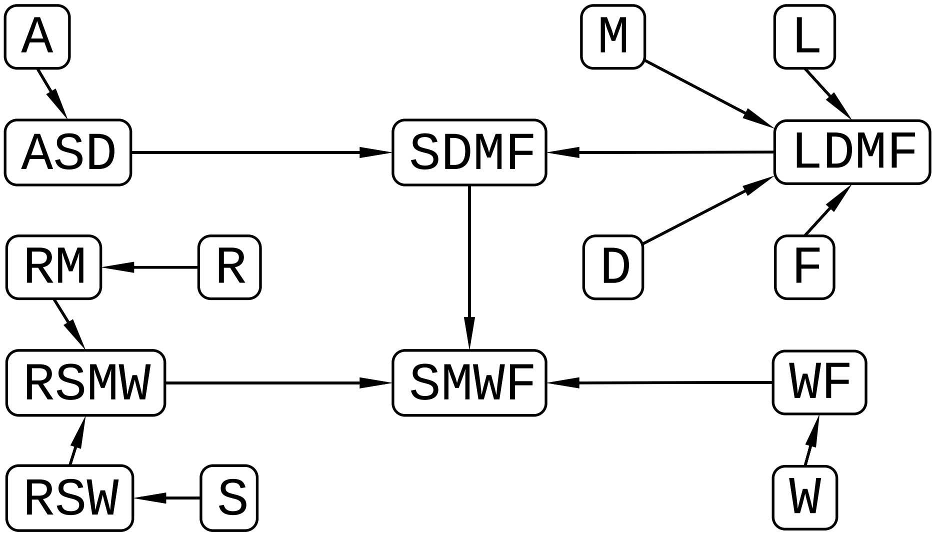


Figure 5. Propagating Inwards

Let us illustrate this with an example. Figure (5) shows the tree of cliques propagating their messages inwards, towards the root $\{S, M, W, F\}$. When $\{S, M, W, F\}$ receives all of its incoming messages, outward propagation can occur. Figure (6) shows the tree of cliques propagating their messages outwards, toward the leaves.

Incidentally, there is nothing in this discussion which is specific to belief functions, belief functions only provide a convenient uniform notation for both the local information and the messages. This passing messages in a tree of cliques (cliques) works equally well in the special case when all the belief functions are Bayesian probability distributions.[1]

[1] People familiar with the work of Lauritzen and Spiegelhalter can see that our methods are exactly equivalent to theirs. Their "potential" representation is the same as the local belief functions we specify; their "set chain" representation (over a collection of *cliques* which could easily be group into a tree) is obtained by passing messages in one direction through the tree of cliques; and

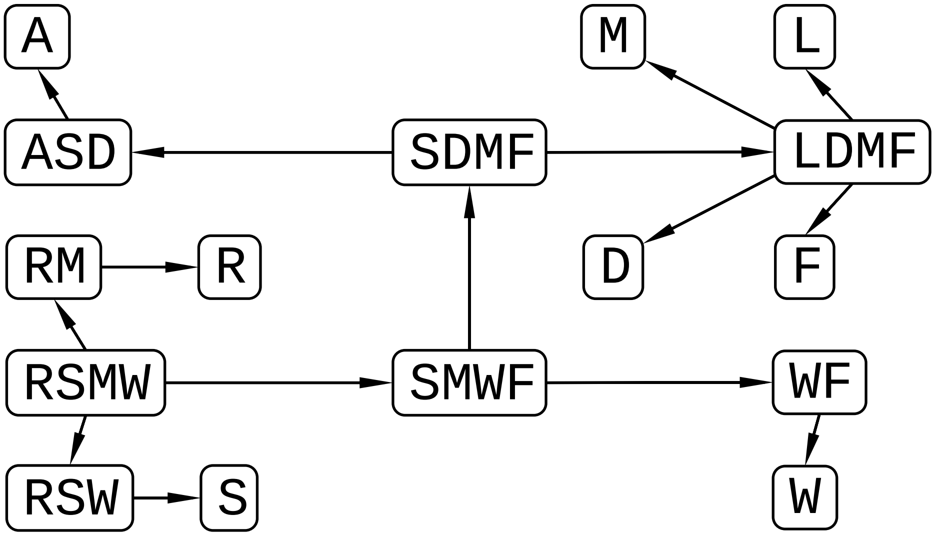


Figure 6. Propagating Outwards

6. Fusion

Now let us turn to the details of what happens inside each clique to “Fuse” incoming messages to produce the outgoing messages. Let us follow, for the moment, a single node, C^*, which has neighbors, $C^1, \ldots, C^k$. We will assume it has received messages from all but the last of its neighbors, that is from $C^1, \ldots, C^{k-1}$. This is shown in figure (7).

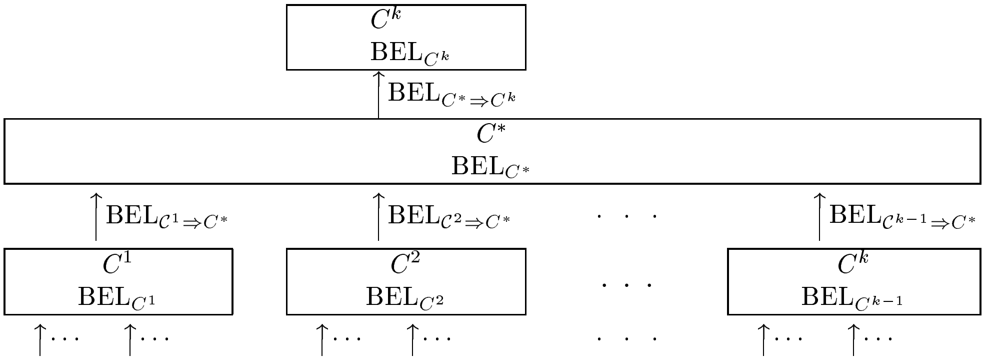


Figure 7. Messages passed to and from node C^*

We now must compute the message, $\mathrm{BEL}_{C^* \Rightarrow C^k}$, that is to be passed to the remaining clique. That is given in equation (1).

$$\mathrm{BEL}_{C^* \Rightarrow C^t} = \left[\mathrm{BEL}_{C^*} \oplus \left(\bigoplus_{s=1}^{k-1} \mathrm{BEL}_{C^s \Rightarrow C^*} \right) \right]_{\rightarrow C^k} \qquad (1)$$

Thus the message passed is just the sum of the incoming messages from all of the other cliques, $\mathrm{BEL}_{C^s \Rightarrow C^*}$’s, with the local information, BEL_{C^*} stored at that clique.

their “marginal” representation (equivalent to our margins of the total belief function) is found by passing messages in the other direction. Thus the reason that they must go through two representations to go from potentials to marginals is because they must pass information in both directions in the tree of cliques.

Here we note that we can do each of these calculations over the frame corresponding to C^*, and then project the result onto the frame corresponding to C^k. [Note: this is where we get the computational cost $\leq m \cdot k \cdot 2^{|\Theta_{C^*}|}$ given in Section 1.] We also note, that a clique can pass messages as soon as it has received messages from all but one of its neighbors. Thus in particular, we can immediately pass message from the leaves, which only have one neighbor. Therefore we can:

- Starting at the leaves, propagate inwards until we reach the root. At each successive stage, the inner nodes receive all of their incoming messages and can pass towards the center.
- At this point, the root has received messages from each of its neighbors, thus it can pass outwards in all directions, calculating its messages by equation (1); each time a clique calculates a message, it omits the destination from the sum. When a clique receives a message from the center, it now has all of its incoming messages and can send outwards to each of its more outward neighbors. This is continued until the leaves receive their messages.

At this point each of the cliques has a series of incoming messages describing the contribution of the other parts of the tree to the total belief function. If we wish to view the margin of the total belief function, $\mathrm{BEL}_{\mathcal{G}}$, corresponding to a given clique, C^*, we simply sum all of the incoming messages with the local components. This is given in equation 2.

$$\mathrm{BEL}_{\mathcal{G}\downarrow C^*} = \mathrm{BEL}_{C^*} \oplus \left(\bigoplus_{s=1}^{k} \mathrm{BEL}_{\mathcal{C}^s \Rightarrow C^*} \right) \tag{2}$$

The separation property, which we mentioned in Section 4, assures us that the marginalization we do in each step will not cause us to lose any crucial information. Thus it enables us to do the calculation over smaller frames, at considerable savings in time. That this procedure is correct can be derived from Kong[1986a,1986b].

Finally, a similar procedure can be used for sensitivity analysis. If we modify one of the component belief functions, we simply need to pass a new set of messages outwards from the modified clique in the tree of closures. The messages passed inwards toward the modified clique will be unchanged, and can be re-used. Thus we have the possibility of rapid sensitivity analysis.

One last note, sensitivity to the graphical structure can be tested in this way. If we construct our original graphical model with an edge of a questionable nature; that is one which we do not know if we want. We can effectively remove that edge by setting its component belief function to the vacuous belief function, and thus test the sensitivity of the model to the inclusion of that edge.

Mass	Focal Element
0.04	$\{0\}$
0.07	$\{1\}$
0.16	$\{1, 0\}$
0.04	$\{2\}$
0.01	$\{2, 0\}$
0.12	$\{2, 1\}$
0.09	$\{2, 1, 0\}$
0.02	$\{3\}$
0.02	$\{3, 1\}$
0.06	$\{3, 2\}$
0.04	$\{3, 2, 1\}$
0.10	$\{3, 2, 1, 0\}$
0.01	$\{4\}$
0.01	$\{4, 2\}$
0.03	$\{4, 3\}$
0.03	$\{4, 3, 2\}$
0.07	$\{4, 3, 2, 1\}$
0.04	$\{4, 3, 2, 1, 0\}$
0.01	$\{5, 4, 3, 2\}$
0.01	$\{5, 4, 3, 2, 1\}$
0.001	$\{5, 4, 3, 2, 1, 0\}$
0.	$\{6, 5, 4, 3, 2, 1, 0\}$ $= \Theta$

Table 1. Focal elements on Arrival delay

7. Some Numbers

Let us examine the kind of output that would come from the fusion and propagation algorithm. We will do this for the Captain's Decision problem. Appendix III gives detailed descriptions of the input component belief functions. The first thing we do, is to calculate the total conflict or mass that would be assigned to the null set. In this case it is zero, that tells us that we are drawing our conclusions by re-enforcing positive evidence, rather than ruling out possibilities by contradictions. Let us now look at the output marginal belief functions, in particular let us examine the belief function over *Arrival Delay*.

Recall that *Arrival Delay* has seven possible values, $\{0, 1, 2, 3, 4, 5, 6\} = \Theta$. Belief functions are defined over the power set of the outcome space, so the belief function takes on 128 values. One way of cutting down the information to a manageable size is to only observe the *focal elements*; that is the sets of possible outcomes which are making a positive contribution to our belief (Those elements with a non-zero mass). There are 21 of these, They are shown in table (1).

We can think of the mass numbers as probabilities that we will receive a message telling us that the true outcome will be in the given set (focal element). We can notice a few things from this table, namely that we have relatively small support for any given day, but there is large support for focal elements representing large sets of days.

As you have noticed, the raw focal elements are difficult to interpret. This is generally true for non-binary variables. Let us try to make some meaningful summaries of the results.

First we look at the lower and upper expectations for the arrival time. These are calculated from the focal elements in a manner similar to the way expectations are calculated; we simply use the minimum or the maximum outcome in the focal element respectively. For this example they are:

$$\mathrm{E}^*(A) = 2.388$$
$$\mathrm{E}_*(A) = 0.824$$

That means that on average, the ship will be between 1 and 2 days late.

Another thing we might try is to look at the beliefs and plausibilities for some sets of interest. One group of sets of interest are the singleton sets corresponding to each day. Those are shown in figure (8); the beliefs (lower probabilities) are the solid lines and the plausibilities (upper probabilities) are the dotted line.

Another quantity that could be of interest, is our belief that the ship will arrive before a certain day, or after a certain day. The former can be represented by sets of arrival days. For example, $\{0, 1, 2\}$ would

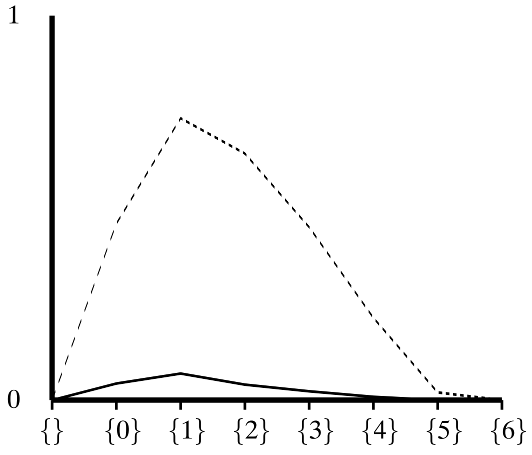


Figure 8. Single Day Beliefs for A̲rrival delay

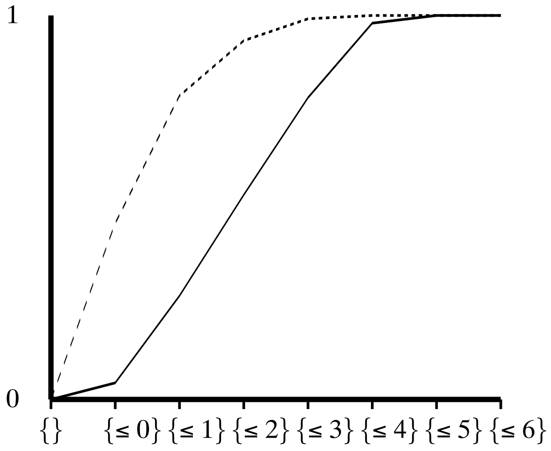


Figure 9a. Fewer than n days for A̲rrival delay

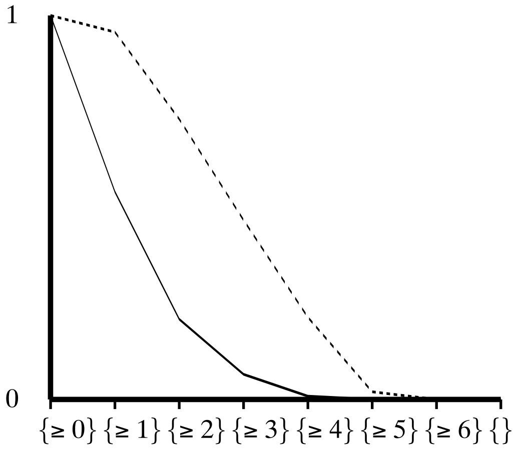


Figure 9b. Greater than n days for A̲rrival delay

represent $<$ 3 days and $\{3, 4, 5, 6\}$ would represent $\geq$ 3 days. These are shown in figures (9a) and (9b). We note that because of the relationships between beliefs and plausibilities, figure 9a is the same as figure 9b when turned upside down and the dotted and solid lines are reversed.

Work on useful summaries is still at a very preliminary stage. The great flexibility that was an advantage when we were specifying the model is now working against us. We are forced to find clever ways of summarizing the information.

8. Conclusions and Future Directions

My experience with both the Captain's example, and the approximately 2,500 lines of lisp code that implement the general algorithms described here, suggests that Graphical Belief Models can successfully model uncertainty in practical decision problems. The algorithms mentioned here are easy to code and to use. Sensitivity analysis is simple in this system, and the theoretical framework is very attractive.

The simple example illustrates how to partition a problem into small pieces using graphical models, and how to model various relationships among the attributes can be modeled. Re-arranging the model hypergraph into a tree of cliques allows us to calculate margins of the composite belief function by a simple message passing algorithm. Finally, we have seen how the simple belief function inputs have given rise to the complex belief function given in table 1. Although we can easily calculate $\mathrm{BEL}(A)$ and $\mathrm{PL}(A)$ for any $A \subseteq \Theta$, we find this belief function is still not easy to comprehend. The methods used in section 7 are only a few of the ways we might interpret these results.

The system has a number of strengths:

1. The graphical modeling is simple to understand. It also forces the user to be precise about the relationships among the attributes. This may seem like a drawback to people who are used to adding rules in a fast and loose manner to a PROLOG database, but actually it forces one to model the interactions correctly in the planning stage, rather than tune them by extensive debugging. Of course our sensitivity analysis methods can be applied for fine tuning or in cases where we are uncertain of the model.
2. The belief functions are a flexible tool for modeling many types of relationships among the variables.
3. The fusion and propagation algorithm allows us to do calculations with much lower computational cost than would otherwise be expected.

Unfortunately, the system also has some weaknesses. For large outcome spaces, a general belief function (such as the one given in table 1) can be a much more complex than a probability distribution over the same space. Furthermore, belief functions are unfamiliar objects and we do not have the wealth of methods for interpretation and anecdotal experience that we do with probabilities. The conflict (the degree to which the belief function in the mode present contradictory evidence) that occurs in assembling the composite belief function is almost certainly a useful tool for discovering what is happening within a graphical model. On the other hand, we have very little experience with evaluating what conflict means. Work on the interpretation of belief functions is just beginning.

Another area where work is just beginning is in the task of evaluating and interpreting the messages. Each message represents the contribution from one branch of the tree, and the conflict from each branch is also available. My experience with these numbers, limited though it is, indicates that they are useful in tracking down the source of surprising results. Once again, our experience in dealing with these sorts of diagnostics is scanty at best.

Finally, if we hope to use this sort of methodology in large problems, we will need to develop the methods further. If we have supercomputing available to us, that should allow us to deal with larger problems. The direct sum operator is very amenable to vectorization possible taking advantage of hypercubic notations for sets, while the message passing algorithm described above seems like it would work well on a variable architecture machine, like a connection machine. Furthermore, just looking at ways of breaking the model hypergraphs into subgraphs that could be dealt with one at a time, and their interface examined last. Thus giving us a modular approach to graphical models.

The next stage in this research is use the system to work a large, real example. This will force us to develop both the theory and the algorithms to accommodate the new example. The methods described here make belief function methodology accessible for practical problems, and its real usefulness has yet to be seen.

Appendix I. Belief Functions

This Appendix provides as a brief review of belief functions (for a more extensive treatment see Shafer[1976]). It also demonstrates the claims for belief functions mentioned in Section 3 of the report.

Let us think of a set of possible outcomes $\Theta = \{\theta_1, \ldots, \theta_n\}$. Now given a subset A of the possible outcomes, we define $\mathrm{BEL}(A)$ as our *belief* (a number between 0 and 1) that the true outcome will be in set A. We define $\mathrm{PL}(A)$, that is the *plausibility* of A, to be $1 - \mathrm{BEL}(\bar{A})$, or one minus the belief in the complement of A with respect to Θ. Furthermore these belief functions are superadditive, that is if $A, B \subset \Theta$ and $A \cap B = \emptyset$ then $\mathrm{BEL}(A) + \mathrm{BEL}(B) \leq \mathrm{BEL}(A \cup B)$. Note that this last rule is a generalization of the corresponding case for probabilities.

With probabilities, one normally thinks of placing a mass function on the possible outcomes. With belief functions the *mass function* must be placed on the power set, 2^Θ, of the outcome space. We normally restrict outselves to belief functions over discrete outcome spaces. The total mass is always one. For a normalized belief function, the mass on the empty set is always zero. (See discussion about the fusion rule below). The mass function is related to the belief function by equation A.1.

$$\mathrm{BEL}(A) = \sum_{B \subseteq A} m(B) \tag{A.1}$$

The mass function of a belief function over a binary outcome space is particularly easy to interpret. For example, suppose we had the outcome space $\Theta = \{\mathtt{F}, \mathtt{T}\}$ where F represents fair weather, and T represents foul weather. Suppose further that we had the following mass function:

$$\begin{aligned} m(\{\mathtt{F}\}) &= 0.6 \\ m(\{\mathtt{T}\}) &= 0.2 \\ m(\Theta) &= 0.2 \end{aligned} \tag{A.2}$$

We can interpret this as either: (1) There's a 20% chance of bad weather, a 60% chance of fair weather and 20% chance of unpredictable weather, or (2) There's a 20–40% chance of bad weather. In terms of betting odds, by this belief function I would be comfortable betting with odds of 1:4 or better that there will be foul weather, or betting against foul weather with odds of 3:2 or better. I am indifferent to all bets within that region. It is useful to think of the mass placed on a given *focal element* (that is a subset of the outcome space which has non-zero mass) as the weight of evidence that suggest the outcome will be in the focal element, that cannot be divided (because of our ignorance) into finer divisions. Shafer, likes to think of receiving a random message, that tells you with a given probability that the outcome will be in the given set. As another example, we might think of the belief function given in (A.2) as an urn containing black balls and white balls. With probability 0.2 we draw a black ball, with probability 0.6 we draw a white ball. With probability 0.2 we draw a ball which looks grey in this light, and we cannot determine its color without further experiments.

Let us examine how some of the different types of relationships mentioned in item 2 of Section 3 can be constructed using belief functions. *Bayesian Probabilities* are simple; we simply assign all of the mass to the *atoms*, or the singleton elements. The mass function then just becomes the p.f. of the distribution. *Ignorance* is also simple; we simply assign mass one to Θ. This special belief function comes up frequently, and is referred to as the *vacuous belief functions*. This states that all we know is that there will be an outcome (one of the ones in our outcome space). If we have observed the *outcome of an experiment*, we can express that as a belief function with mass one on the atom corresponding to the observed outcome.

Logical Rules are a bit more complex, and deserve a slightly more complex treatment. Before I go on, I want to introduce the notion of a *multivariate belief function*. Two define a belief function over two, or more variables, we simply define a univariate belief function over the product space. Thus if we wished to define a belief function over both *W̲eather* and *F̲orecast*, with the corresponding outcome spaces, Θ_W and Θ_F, we would define a belief function over the product space $\Theta = \Theta_W \times \Theta_F = \{(\mathtt{F},\mathtt{F}), (\mathtt{F},\mathtt{T}), (\mathtt{T},\mathtt{F}), (\mathtt{T},\mathtt{T})\}$.

Now consider the logical proposition, "The Forecast and the actual Weather agree", or $W \Leftrightarrow F$. This can be represented by its truth table, $\{(\mathtt{F},\mathtt{F}), (\mathtt{T},\mathtt{T})\}$. Thus a belief function that expressed the relationship $W \Leftrightarrow F$ could be constructed by placing mass one on the set $\{(\mathtt{F},\mathtt{F}), (\mathtt{T},\mathtt{T})\}$.

Although the technique I have illustrated with the previous example is generally useful in a large number of cases, it is not completely appropriate to the example. Most people, even weathermen, would not have complete confidence (corresponding to mass one) in the proposition that the actual *Weather* will follow the *Forecast*. What we do, is apply a trick from Shafer's book, and construct a *simple support function* for the logical proposition in question. Thus we might asses that the accuracy of the weather service in predicting storms is 80%; that is with probability 0.8, $W \Leftrightarrow F$, and with probability 0.2, the weather service is out to lunch. Thus we construct a belief function by placing mass 0.8 on the truth table, $\{(\mathtt{F},\mathtt{F}),(\mathtt{T},\mathtt{T})\}$ and placing mass 0.2 on the entire frame, Θ. Mass on Θ, indicates our belief that the weather service occasionally makes mistakes, and that those mistakes are in unpredictable directions. Note: if we placed mass one on the logical proposition, that would become a *total support function* for that proposition.

Let us look at another example, this time taken from the MYCIN expert system.[2] They use the following rule:

```
IF: 1) The stain of the organism is gram positive, and
    2) The morphology of the organism is coccus, and, and
    3) The growth conformation of the organism is chains
THEN:  There is suggestive evidence (.7) that the identity of the or-
       ganism is streptococcus.
```

This example seems to suggest a simple support function over a multivariate belief function. The frame would correspond to the product space of the four attributes, *stain*, *morphology*, *growth*, and *identity*. We could construct a truth table from the logical rule, and assign it mass of 0.7. The experts consulted by the MYCIN team didn't seem to know what to do with the remaining mass, but they felt very uncomfortable with assigning it to the proposition that the organism was not streptococcus (the probabilistic alternative). Therefore, the MYCIN system of confidence factors was developed. With belief functions, we can assign the remaining mass to Θ, corresponding to the proposition with 30% probability the information about the organism tells us nothing about the identity of the organism, perhaps because some mistake was made in the experimental procedure, or because of non-specificity of the tests. Therefore, our *simple support functions* implement the MYCIN *confidence factors*.[3]

Let us further examine the concept of multivariate belief functions. In the kind of problem we described in sections 1 and 2, the proper outcome space would be the product space over all of the "attributes" (or variables) of interest. Obviously, this belief function is large, with $2^{|\Theta|}$ possible focal elements. But our graphical modeling trick, allows us to think of smaller margins of the outcome space. Such smaller margins are called *frames of discernment*. Thus $\Theta_W \times \Theta_F$ is the frame of discernment corresponding to the two attributes W and F.

Using belief functions it is very simple to move between smaller and larger frames. To move from a large frame to a smaller frame, you simply project each focal element onto the smaller frame. This produces a new mass function for the belief function over the smaller frame. Thus to *marginalize* the belief function BEL_Θ over the frame Θ, to a smaller frame Θ_C, you follow equation A.2.

$$m_{\Theta\downarrow\Theta_C}(A) = \sum_{\substack{B\subseteq\Theta \\ B\cap\Theta_C=A}} m_\Theta(B) \qquad \forall A \subseteq \Theta_C \tag{A.3}$$

We call the induced belief function $\mathrm{BEL}_{\Theta\downarrow\Theta_C}$ or just $\mathrm{BEL}_{\downarrow C}$. Note that if we factor $\Theta = \Theta_C \times \Theta'_C$ then $\mathrm{BEL}_{\Theta\downarrow\Theta_C}(A) = \mathrm{BEL}_\Theta(A \times \Theta'_C)$. Note also, that this marginalization is exactly analogous to the marginalization of multivariate probability distributions.

Unlike probabilities, we can also easily go in the opposite direction. This process is called *minimal* or *vacuous extension*. Let us consider going from a frame Θ_C to a frame $\Theta_B = \Theta_C \times \Theta'_C$. What we would like is a belief function that is defined over Θ_B, yet its margin when projected on Θ_C is the original belief function. What we do is we simply extend the focal elements by crossing them with Θ'_C. Thus $m_{\Theta_C\uparrow\Theta_B}(A \times \Theta'_C) = m_{\Theta_C}(A)$ for all $A \subseteq C$ and the mass is defined as zero elsewhere. You can think of this process as using a vacuous belief function to extend the old belief function over a larger frame.

We can think of both marginalization and vacuous extension as set operation on belief functions. In both cases, the focal elements are expanded, or contracted as necessary to fit the new frame. In the case of marginalization, certain focal elements may become identical, so these focal elements are summed over,

[2] Buchanan and Shortlife[1984], pp. 238–239.

[3] Although we can represent rules with confidence factors we do not use the same rule of combination. Our rule of combination is explained below.

collapsing the original belief function. One can also combine the operations of marginalization and minimal extension, to an operator known as projection. Thus $\mathrm{BEL}_{B\to C}$ would be a belief function defined over Θ_B, that was marginalized to $\Theta_{B\cap C}$ and minimally extended to Θ_C.

Let us turn to our method of combining belief functions. This operation, sometimes called the *direct sum*, the *orthogonal sum* or *Dempster's rule of combination* is usually written as $\oplus$. It is used for combining two "independent"[4] belief functions over the same frame (thus the projection operator often must be applied before the direct sum). The direct sum is defined in terms of the mass and is given in equation A.4.

$$\mathrm{BEL}_1 \oplus \mathrm{BEL}_1(C) = \sum_{\substack{A,B\subseteq\Theta \\ A\cap B=C}} m_1(A)\cdot m_2(B) \tag{A.4}$$

Notice that this process could put non-zero mass on the empty set. This mass on the empty set is called κ, and is referred to as the *conflict*. The belief function is then renormalized, by dividing by $1-\kappa$. Computationally, there is no reason to renormalize every time we add two belief functions, we can save the renormalization step until the very end, simply carrying along the unnormalized belief functions and treating them the same way we would ordinary belief functions.

The computational cost for the direct sum operator is the product of the number of focal elements in each of the two belief functions to be combined. A simple computation shows that the worst case is if both belief functions have as many observations as there are elements in the power set of the outcome space. Thus we have a worst case computational cost of $2\cdot 2^{|\Theta|}$, calculations. Fortunately, usually our graphical models are constructed out of simple components, so we usually do much better than this. Still, one can see the value of combining over small frames, the way the fusion and propagation algorithm works, rather than over the whole space.

If we look carefully at equation A.4 we can see two special cases. If both belief functions have a single focal element (with mass one), we can see that the result will also be a belief function with a single focal element, which will be the intersection of the other two belief function's focal elements. Thus for logical belief functions, we re-produce logical inference, which in truth table notation, is done by intersecting truth tables. In the special case where BEL_2 is a Bayesian belief function, (i.e. it represents a probability distribution) and BEL_1 represents a conditional probability distribution (see the discussion below), we can reguard them as a "prior" and a "likelihood" respectively. A close look at the conflict, reveils that $1-\kappa$ is simply the sum over all possible outcomes, thus it is the normalization constant of Bayes Theorem. Thus Dempster's rule is a more general case of Bayes Theorm.

Let us take a closer look at the conflict, which is an important diagnostic tool. When the conflict is high (close to 1), then our component belief functions have a high degree of disagreement. We are hence mostly drawing our conclusions mainly by eliminating conflicting possibilities. On the other hand, low conflict could indicate lack of specificity in the model; it could mean that all of the component belief functions are two vague, or it could simply have arisen as an artifact of the model being built out of conditional belief functions. Similarly, high conflict arises naturally when you are combining Bayesian belief functions, thus to what extent high conflict is bad, we don't know. We have very little experience to rely on here, and as with all diagnostic tools, time will tell.

One artifact of leaving the normalization step for last, is that the messages passed between nodes of the tree of cliques are unnormalized belief functions. Thus it is possible to define the *conflict from a branch* of the tree of cliques. This promises to be a useful tool in tracking down areas of high conflict within a graphical belief function, but again we have little experience with it.

Now let us turn to the notion of a "conditional belief function." Let C be a collection of attributes, and $A\in C$. Further suppose that we have a belief function over the frame corresponding to C, BEL_C and we wish to condition it on the fact that $A=a_0$, for some value of the attribute. If we let C' be the attributes of C other than A, then we can create a belief function with the single focal element $a_0\times\Theta_{C'}$ which corresponds to the observation $A=a_0$; call that belief function, $\mathrm{BEL}_{A=a_0}$. Conditioning BEL_C on $A=a_0$ corresponds

[4] The notion of independence of belief functions is not very well understood. We tend to think of it in the same way that we do statistical independence, but that relies on the rather backwards definition that two belief functions (or distributions) are independent if they factor. When we run into belief functions that are dependent, or possibly dependent, we usually break the problem up by assigning unique attribute labels to each of the attributes that are shared in common to the two dependent belief functions, and then explicitly modeling the dependence with a third belief function over both versions of the common attributes. These three belief functions can be regarded as independent while the original first two could not.

to $\mathrm{BEL}_C \oplus \mathrm{BEL}_{A=a_0}$. We similarly can condition on $a_1, \ldots, a_n$. Thus to create a conditional belief function, it is sufficient to find a belief function that satisfies the n equations:

$$\begin{aligned}
\mathrm{BEL}_C(B) \oplus \mathrm{BEL}_{A=a_0}(B) &= \mathrm{BEL}(B \mid A = a_0) \\
\mathrm{BEL}_C(B) \oplus \mathrm{BEL}_{A=a_1}(B) &= \mathrm{BEL}(B \mid A = a_1) \\
&\vdots \\
\mathrm{BEL}_C(B) \oplus \mathrm{BEL}_{A=a_n}(B) &= \mathrm{BEL}(B \mid A = a_n)
\end{aligned} \tag{A.5}$$

Thus our conditional belief functions are just ordinary belief functions. This makes belief functions a "potential" representation, similar to the one used in log-linear modeling and gives us considerable computational convenience.[5]

There is only one problem, the solution to equation A.5 is not unique. Even if we were to further restrict the belief function by forcing the margin on A to be vacuous, it still would not be unique. Thus we need some other method of selecting the correct conditional belief function. One trick that is used is conditional embedding (Shafer[1982]). Here each one of the conditional belief functions from equation A.5 is assumed to come from a separate body of knowledge. (For example our experiences with breakdowns at sea given we did, or did not perform maintenance at the dock, are somehow reguarded as two separate groups of experiences. Analogously to the way we vacuously extended the multivariate belief functions, we vacuously extend each of the RHS's of equation A.5 for the case when $A \neq a_i$. We then combine them using Dempster's rule. Of course this relies heavily on the independence assumption to work, and it must be applied with care. However, in the special case where we are trying to simulate a Bayesian conditional distribution, it works quite well.

At this point we have methods for creating each of the component belief functions in our model. The *global* or *graphical* belief function, denoted $\mathrm{BEL}_{\mathcal{G}}$ is simply the direct sum of all of the components. The games we play with the tree of cliques, merely help us to partition these calculations into calculations done in smaller frames, thus decreasing the worst case cost.

Appendix II. The Tree of Cliques

The idea for the tree of cliques arises naturally from Kong's 1986 thesis. Kong proposes that the margin of a graphical belief function can be calculated by successively removing attributes from the graphical model. Each time an attribute is removed, belief functions are "fused" over the closure of the removed attribute, and a new belief function is created representing that information. That belief function is marginalized to the neighborhood of the removed attribute (that is summed over the remove attribute), and the resulting information is then represented as a new graphical model, with one fewer attributes, and a new edge representing the combination of all the edges containing the deleted attribute. This step can be though of as passing messages into the interior of the graph.

From the preceding paragraph, it should be obvious that there is a strong connection between the methods in Kong[1986a] and the fusion and propagation algorithm given in Dempster and Kong[1988] and here. Based on this connection, Kong developed an algorithm for building a tree of cliques (Kong[1986b]). I present it here along with some extensions.

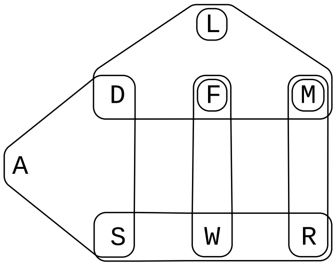
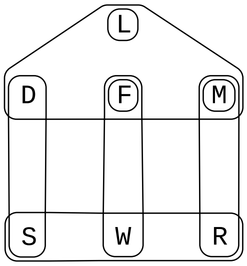
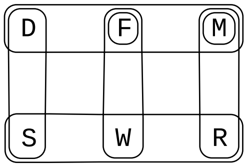
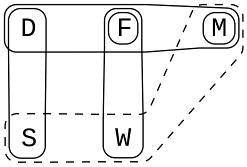
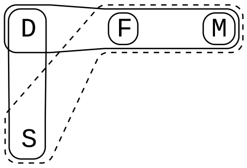
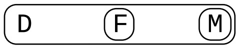
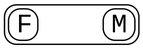
Kong called the deletions with replacement that occurred in his thesis, *r-deletion*. R-deleting an attribute A from a hypergraph $\mathcal{G}$ is done by: (1) removing A and all edges containing A from $\mathcal{G}$, and (2) adding to the resulting hypergraph $\mathrm{N}(A \mid \mathcal{G})$ (the neighborhood of A in $\mathcal{G}$ as an edge). Notice that r-deletion preserves connectivity in our model hypergraph; that is if we marginalize over A, then the neighbors of A become dependent. This property allows Markov conditions to hold in the tree of cliques. To construct the tree of cliques, we r-delete all of the nodes of the model graph, and construct a tree from the closures of the deleted points at each step of the process (recall from the first paragraph of this section how it was the closure of the deleted point over which we needed to combine belief functions).

Now let us construct a tree of cliques for the Captain's Decision Problem. We start with a *deletion order*, indicating the order in which we will remove the attributes. (This deletion order is not unique and will, in general, determine the quality of the tree of cliques produced. I shall discuss choosing an optimal deletion order shortly). If we let $A_n, \ldots, A_1$ be the deletion order, and if we let $\mathcal{G}_n = \mathcal{G}$ be our original model hypergraph, then our r-deletions will create a series of successively smaller hypergraphs, $\mathcal{G}_n, \ldots, \mathcal{G}_1$. Table II-1, shows those hypergraph, along with the neighbors and closures for the r-deleted points for the deletion order A, L, R, W, S, D, M, F.

[5] For those of you familiar with the Lauritzen and Spiegelhalter paper, contrast it with there results.

Table II-1 R-deletions for the Captain's Decision

For each stage i, of the reduction of the hypergraph, this table shows the successively smaller versions of the hypergraph $\mathcal{G}_i$ from which the attribute A_i is removed. The table also shows the neighbors of A_i in $\mathcal{G}_i$, $\mathrm{N}(A_i \mid \mathcal{G}_i)$, which is added as an edge to the hypergraph $\mathcal{G}_{i-1}$ as part of the r-deletion process, and the closure of A_i in $\mathcal{G}_i$, $\mathrm{Cl}(A_i \mid \mathcal{G}_i)$, which will become the nodes in the tree of cliques.

i	$\mathcal{G}_i$	A_i	$\mathrm{N}(A_i \mid \mathcal{G}_i)$	$\mathrm{Cl}(A_i \mid \mathcal{G}_i)$
8		A	$\{D, S\}$	$\{A, D, S\}$
7		L	$\{D, F, M\}$	$\{L, D, F.M\}$
6		R	$\{S, W, M\}$	$\{R, S, W, M\}$
5		W	$\{S, F, M\}$	$\{W, S, F, M\}$
4		S	$\{D, F, M\}$	$\{S, D, F, M\}$
3		D	$\{F, M\}$	$\{D, F, M\}$
2		F	$\{M\}$	$\{F, M\}$
1		M	$\{\}$	$\{M\}$

From the list of closures and neighborhoods we have formed, we build the tree of cliques. Going in the reverse order, we add the $\mathrm{CL}(A_i \mid \mathcal{G}_i)$ into the tree of cliques at each step. There are two different cases: (1) The neighborhood of A_i is already a node in the tree of cliques. In this case, we relabel that node to be the closure of A_i. This happens here in the first, second, third and fourth steps, and produces the single node $\{S, D, F, M\}$. (2) The neighborhood of A_i is a subset of some node in the tree of cliques. In this case, we add the closure of A_i as a new node to the tree, and connect it to any node which contains the neighborhood of A_i. Figure II-1 shows the tree of cliques for the Captain's Decision, numbering them at the step in which they appear.

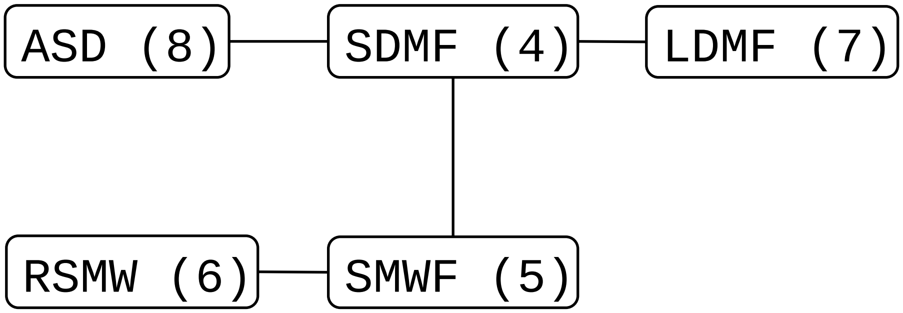


Figure II-1. Unaugmented Tree of Cliques

This procedure will produce a tree of cliques from any hypergraph. A simple inductive proof will verify that such a tree always exists, although it is not unique. To minimize computational costs, I always try to connect new nodes to the smallest possible node. A simple inductive proof also shows that the tree of cliques (the tree produced by this method) has the separation property. Both of these proofs are given in Kong[1986b]. We also note that steps 6 and 5 of the r-deletion process create two new edges, $\{S, W, M\}$ and $\{S, F, M\}$ (shown in $\mathcal{G}_5$ and $\mathcal{G}_4$ respectively as dotted lines). Thus we can consider a graph that is made of a union of all of the graph from our deletion process $\bigcup \mathcal{G}_i$. This graph will have the additional edges created during r-deletion "filled-in", and the resulting graph will be triangulated (acyclic). Thus our procedure is equivalent to a fill-in procedure on the model hypergraph, and the nodes in our tree are "cliques" of the filled-in graph. This is how the tree of cliques gets its name.

Notice that the tree of cliques given in figure II-1 does not contain many of the nodes found in the complete tree of cliques (figure 4). In particular, it is missing most of the nodes corresponding to the original edges of the model hypergraph. On the other hand, every edge of the model hypergraph is contained in at least one node of the tree of cliques. So, the belief function one should associate with each node is the combination of all of the edges which fall within that node. In practice, it is easier to augment the tree of cliques by adding the original edges to the tree. Each edge of the model hypergraph becomes a node in the tree and is connected to any node already in the tree which contains it. The singly augmented tree of cliques is shown in figure II-2.

Notice that some margins of interest are not represented in the Singly Augmented Tree of Cliques. In particular, the node representing the margin $\{A\}$ is not in the graph (although it could be deduced from $\{A, D, S\}$. Furthermore, we may wish to condition on a delayed arrival (that is introduce a belief function fixing A at some value) and try and determine the most probable causes. Thus it might be useful to include the margin $\{A\}$ in our analysis. Thus we do a second augmentation step in which we add it the singleton nodes which we feel are of interest. This tree is shown in figure 4. [Note: We cannot add an arbitrary combination of attributes to the tree, any combination we add must be a subset of one of the existing nodes

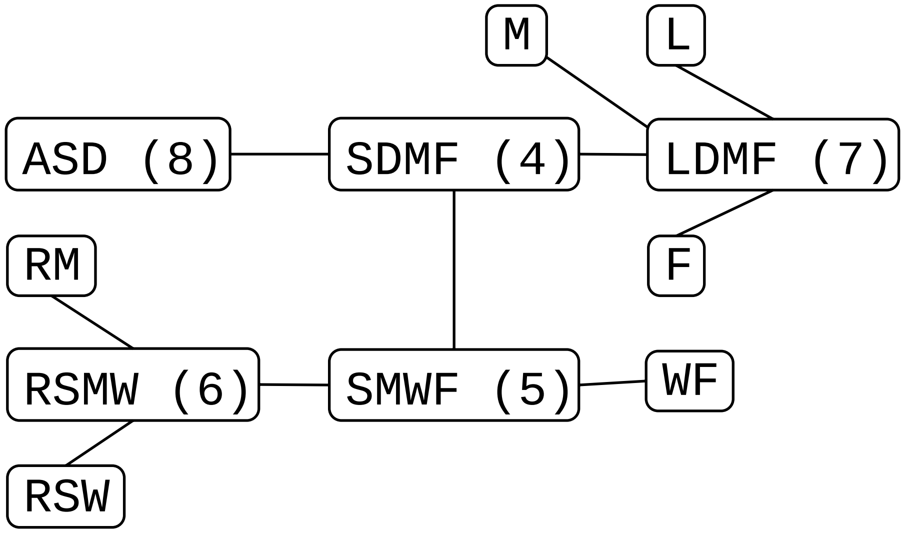


Figure II-2. Singly Augmented Tree of Cliques

of the tree. If we wish to examine a margin we do not find in the tree, we must go back to the original model hypergraph and introduce an edge containing the attributes in the margin we wish to examine (we may give that edge a vacuous belief function) and recompute the tree of cliques.]

Optimal Deletion Order

Remember that to build the tree of cliques we noted that we needed a deletion order, but made no remarks were made on how to select one. In general, the computational cost associated with the tree of cliques is proportional to the size of the largest node in the tree. The size of the largest node in the tree, in turn, is strongly effected by the deletion order, and so finding a good deletion order is critical to finding an efficient tree. Because the problem is related to the problem of finding an optimal fill-in, which is known to be NP-hard (Yannakakis[1981]), Kong and I have conjectured that this problem is NP-hard as well. Instead, we have developed heuristics for producing a “good” (although not optimal) tree of cliques.

Kong[1986a] suggests the following procedure which he calls the “one step look ahead”:

At each step of the r-deletion procedure, look at the graph $\mathcal{G}_i$.

1) If there exists a leaf (that is an attribute, A such that $\mathrm{N}(A \mid \mathcal{G}_i)$ are all connected), delete that leaf. (Kong[1986a] has shown that when it can be done, this is always an optimal strategy).
2) If there is no leaf, find all attributes, A such that the size of $\mathrm{CL}(A \mid \mathcal{G}_i)$ is as small as possible.
3) If there still exists more than one such node, break ties arbitrarily.

In particular, at the second step of the one step look ahead procedure, we see the criterion is to find the point which when deleted produces the smallest clique. Therefore, we will refer to this procedure as the *one-sc* procedure (for one step, smallest clique). We could consider other one step look ahead methods which use a different criteria in the fourth step. In particular if we use instead of (2):

2a) Find all attributes, A such that the fewest number of simple edges need to be filled-in in order to make $\mathrm{CL}(A \mid \mathcal{G}_i)$ a clique (completely connected).

This procedure is called *one-ff* for one step fewest fill-ins.

We also note that in criterion (3) we are breaking ties arbitrarily. If instead we used one of the other candidate criterion, we would get “one and a half step look ahead” procedures. In particular by using (1), (2), (2a) and then (3) we have *one-sc-tff* or one step smallest cliques, break ties with fewest fill-ins. By reversing the order of (2) and (2a) we have *one-ff-tsc* or one step fewest fill-ins, break ties with smallest cliques.

Lauritzen and Spiegelhalter [1988] recommend using a procedure called maximum cardinality search (*mc-search*), from a paper by Tarjan and Yannakakis [1984]. Maximum cardinality search provides a fast mechanism for checking to see whether or not a model hypergraph is triangulated, however, the tree of cliques produced from the deletion ordering produced will in general have no optimality properties. An earlier paper by Rose, Tarjan and Lueker[1976], discuses a procedure called lexicographic search (*lex-search*), which produces a “minimal” although not minimum, fill-in. Although both these procedures are designed to minimize fill-ins, not the size of the largest cliques, they are being used to create trees of cliques from model hypergraphs. We will compare these methods with the one step look ahead methods described above.

First, let us look at the time that is needed to find the deletion order. From Rose *et al.* we know that lex-search runs in $O(ne)$ time, where n is the number of nodes and e is the number of edges. From Tarjan and Yannakakis we know that mc-search runs in $O(n+e)$ time. Each of the one-step procedures will run in $O(n^2e)$ time [one loop for each node deleted, and one for each node searched, calculating the clique size or the the fill-in is an $O(e)$ operation.

We will try this procedure out on four different graphs. The first is the Captain’s problem given in this paper. The second is from Rose et al. and is given in figure II-3. Note that the numbering of the nodes is given by the lexicographic search algorithm.

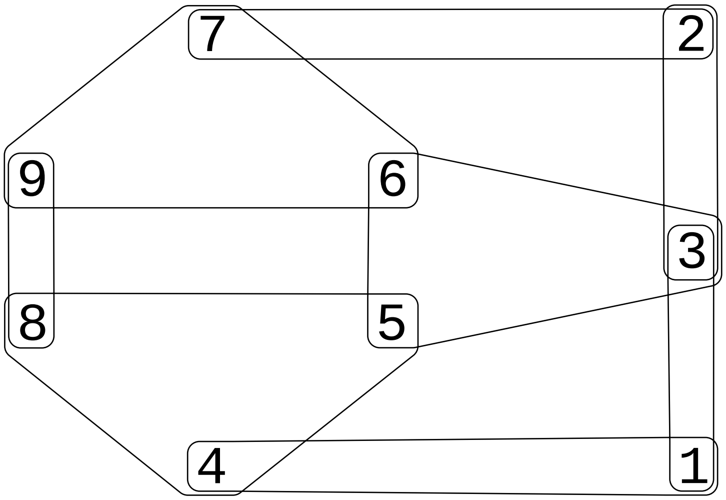


Figure II-3. Graph from Rose et al.

The last two graphs are square lattices which are known to be difficult cases. We will use both the 4×4 and 5×5 square lattice (shown in figures II-4 and II-5 respectively).

If we assume for the moment that each node of our model hypergraph represents a binary attribute, then we find that the upper bound on the computational cost for a given tree of cliques is $\sum_{j=1}^{m} 2^{2^{n_j}}$ (where n_j is the size of the jth clique). Another way of looking at the computational cost is simply to examine the clique sizes directly. If we order the cliques according to size, then we could say that a deletion order whose clique sizes are n_j for $j=1,\ldots,m$ is better than a deletion order whose clique sizes are n'_j for $j=1,\ldots,m'$, if there exists a k such that $n_j = n'_j$ for $j<k$ and $n_k < n'_k$, or if $m < m'$ and $n_j = n'_j$ for $j \le m$. Table II-2 gives the sizes of all the cliques produced by using each of the methods described above to obtain a deletion order.

Even though the maximum cardinality search and the lexicographic search are faster than the one step look ahead algorithms, the trees they produce are less cost effective, even on the simple example of the Rose *et al.* graph. Among the one step look ahead algorithms, *one-ff*, *one-ff-tsc* and *one-sc-tff* all seem to preform equally well. Intuitively, reducing unnecessary fill-ins, would lead to smaller cliques at later stages of the process, so this result is not surprising. Also, smaller cliques will naturally require fewer fill-ins. Among all of the methods, the heuristic for the *one-sc-tff* method is the theoreticly strongest, but I am conjecturing that the *one-ff* method will always do as well. We note that the *one-sc* method of Kong[1986a], although it does

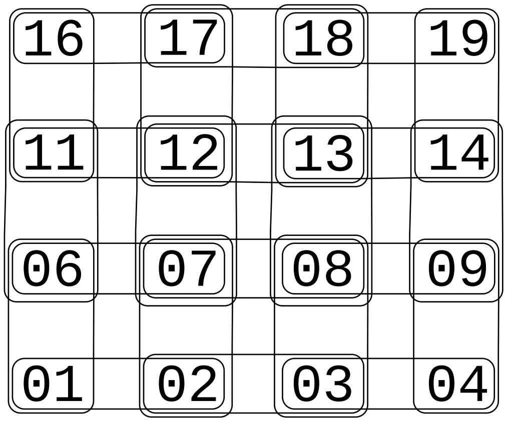


Figure II-4. 4×4 *Square Lattice*

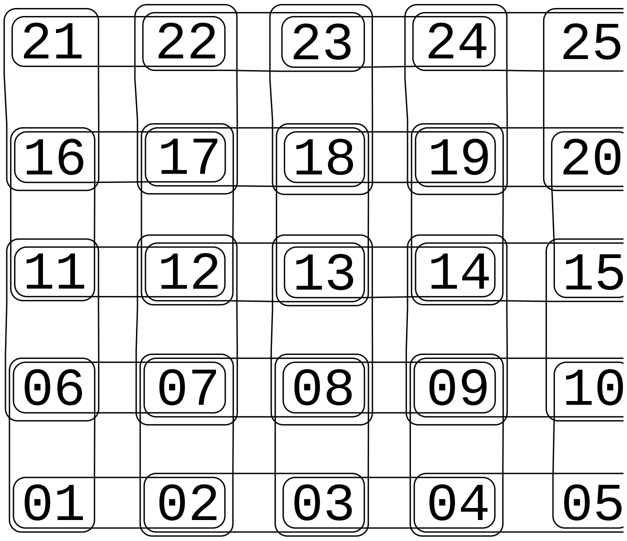


Figure II-5. 5×5 *Square Lattice*

Method	Captain	Rose *et al.*	4×4	5×5
mc-search	5, 4×2, 3	5×2, 4, 3×3	5×9, 4, 3	9×3, 8×2, 7, 6×2, 4×4, 3×4
lex-search	4×4, 3	5×3, 4, 3	5×6, 4×4, 3×2	7×4, 6×2, 5, 4×8, 3×4
one-sc	4×4, 3	4×4, 3×2	5×4, 4×4, 3×4	6×3, 5×9, 4×4, 3×4
one-ff	4×4, 3	4×4, 3×2	5×4, 4×4, 3×4	6×3, 5×6, 4×7, 3×4
one-sc-tff	4×4, 3	4×4, 3×2	5×4, 4×4, 3×4	6×3, 5×6, 4×7, 3×4
one-ff-tsc	4×4, 3	4×4, 3×2	5×4, 4×4, 3×4	6×3, 5×6, 4×7, 3×4
arbitrary	5, 4×2, 3	5×4, 4	5×9, 4, 3	13, 12, 11, 10, 9×4, 8, 7×3, 5

Table II-2. Clique sizes for various deletion orders on various graphs. Clique sizes are given by size '×' repetition.

not do as well as some of the modifications and extensions of that method, does in fact perform equally well with the other methods in Kong's original criterion, the size of the largest clique.

For reference, I include table II-3 which shows the deletion order produced by each procedure. Note that the "arbitrary" order is the one used by all of the other procedures to break ties.

Method	Captain	Rose *et al.*	4×4	5×5
mc-search	A, L, F, W, M, R, D, S	1, 3, 2, 5, 4, 6, 7, 8, 9	1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19	1, 5, 2, 4, 3, 21, 6, 16, 11, 25, 10, 20, 15, 22, 24, 23, 7, 9, 8, 17, 12, 19, 14, 18, 13
lex-search	L, F, M, W, R, D, S, A	1, 2, 3, 4, 5, 6, 7, 8, 9	1, 2, 6, 3, 7, 11, 4, 8, 12, 16, 9, 13, 17, 14, 18, 19	1, 5, 21, 25, 2, 6, 4, 10, 16, 22, 20, 24, 3, 7, 11, 9, 15, 23, 17, 19, 8, 12, 14, 18, 13
one-sc	A, L, S, R, D, M, W, F	3, 1, 9, 7, 6, 8, 5, 4, 2	19, 16, 4, 1, 18, 14, 11, 3, 17, 13, 12, 9, 8, 7, 6, 2	25, 21, 5, 1, 23, 15, 11, 3, 13, 19, 17, 9, 7, 24, 20, 16, 4, 23, 18, 14, 12, 8, 10, 6, 2
one-ff	A, L, S, R, D, M, W, F	3, 1, 9, 7, 6, 8, 5, 4, 2	19, 16, 4, 1, 18, 14, 11, 3, 7, 17, 13, 12, 9, 8, 6, 2	25, 21, 5, 1, 24, 16, 4, 15, 22, 6, 2, 19, 17, 20, 13, 9, 7, 11, 12, 14, 8, 3, 10, 18, 23
one-sc-tff	A, L, S, R, D, M, W, F	3, 1, 9, 7, 6, 8, 5, 4, 2	19, 16, 4, 1, 18, 14, 11, 3, 7, 17, 13, 12, 9, 8, 6, 2	25, 21, 5, 1, 24, 16, 4, 15, 22, 6, 2, 19, 17, 20, 13, 9, 7, 11, 12, 14, 8, 3, 10, 18, 23
one-ff-tsc	A, L, S, R, D, M, W, F	3, 1, 9, 7, 6, 8, 5, 4, 2	19, 16, 4, 1, 18, 14, 11, 3, 7, 17, 13, 12, 9, 8, 6, 2	25, 21, 5, 1, 24, 16, 4, 15, 22, 6, 2, 19, 17, 20, 13, 9, 7, 11, 12, 14, 8, 3, 10, 18, 23
arbitrary	A, S, D, R, M, W, F, L	9, 8, 7, 6, 5, 4, 3, 2, 1	19, 18, 17, 16, 14, 13, 12, 11, 9, 8, 7, 6, 4, 3, 2, 1	13, 18, 14, 12, 8, 19, 17, 9, 7, 23, 15, 11, 3, 24, 22, 20, 16, 10, 6, 4, 2, 15, 21, 5, 1

Table II-3. Deletion orders produced by various methods on various graphs.

Appendix III. The Belief Functions for the Captain's Problem

The following is the promised description of the exact inputs used in the Captain's Decision problem. Let us discuss the belief functions one at a time. We will start with a description of all the attributes, then the multi-attribute edges (relationships between variables) and finish with the singleton edges (priors).

You will recall that the basic description of the attributes was given in Section 1. We will summarize them here in table III-1, along with the range of values for each attribute. We use the notation T for true and F for false.

Let us examine the edge $\{A, D, S\}$ The relationship among the three variables can be expressed by the rule $A = D + S$. Consider the set of all tuples (a, d, s) such that $a = d + s$, where $a \in \{0, \ldots, 6\} \quad d, s \in \{0, \ldots, 3\}$. That set forms a truth table for the addition function. (Recall from set theory that all functions can be expressed as relations). Thus we can characterize the constraint that $A = D + S$ with the logical

Attribute	Description	Θ	Notes
A	*Arrival delay*	$\{0,1,2,3,4,5,6\}$	Days
D	*Departure delay*	$\{0,1,2,3\}$	Days
S	*Sailing delay*	$\{0,1,2,3\}$	Days
L	*Loading*	$\{\mathtt{T},\mathtt{F}\}$	T means delay in loading
F	*Forecast*	$\{\mathtt{T},\mathtt{F}\}$	T means foul weather
M	*Maintenance*	$\{\mathtt{T},\mathtt{F}\}$	T means maintenance performed
W	*Weather*	$\{\mathtt{T},\mathtt{F}\}$	T means foul weather
R	*Repairs*	$\{\mathtt{T},\mathtt{F}\}$	T means repairs made

Table III-1. Attributes for Captain's Decision

belief function:

$$m\left(\left\{\begin{array}{llll}(0,0,0), & (1,0,1), & (2,0,2), & (3,0,3),\\ (1,1,0), & (2,1,1), & (3,1,2), & (4,1,3),\\ (2,2,0), & (3,2,1), & (4,2,2), & (5,2,3),\\ (3,3,0), & (4,3,1), & (5,3,2), & (6,3,3)\end{array}\right\}\right)=1$$

The belief function describing the relationship $\{D,L,M,F\}$ is also a logical one. In this case any of L, M or F being true adds one to the total delay, D. In an analogous way we can build a truth table for this relationship. We then assign the following belief function to that rule:

$$m\left(\left\{\begin{array}{llll}(0,\mathtt{F},\mathtt{F},\mathtt{F}), & (1,\mathtt{T},\mathtt{F},\mathtt{F}), & (1,\mathtt{F},\mathtt{T},\mathtt{F}), & (1,\mathtt{F},\mathtt{F},\mathtt{T}),\\ (2,\mathtt{F},\mathtt{T},\mathtt{T}), & (2,\mathtt{T},\mathtt{F},\mathtt{T}), & (2,\mathtt{T},\mathtt{T},\mathtt{F}), & (3,\mathtt{T},\mathtt{T},\mathtt{T})\end{array}\right\}\right)=1$$

The belief function describing the relationship $\{S,W,R\}$ is very similar, the truth table is very similar. However, we would like to allow for the possibility that there could be delays at sea for causes other than the ones we have listed. Thus instead of using a logical belief function for the proposition, we use a simple support function for the proposition. We assign mass 0.9 to the proposition, and the remaining mass 0.1 to the whole frame. This means we only expect our rule to be accurated 90% of the time.

$$\begin{aligned} m\left(\{(0,\mathtt{F},\mathtt{F}),(1,\mathtt{T},\mathtt{F}),(1,\mathtt{F},\mathtt{T}),(2,\mathtt{T},\mathtt{T})\}\right) &= 0.9\\ m\left(\Theta\right) &= 0.1\end{aligned}$$

If the weatherman was 100% accurate, we could express the rule $\{F,W\}$ as $F \Leftrightarrow W$. Thus once again we can construct the truth table for this logical relation, it is: $\{(T,T),(F,F)\}$. As we discussed in Appendix II, what we really want is a simple support function over that relationship. If we have 80% confidence in the weatherman, that belief function is:

$$\begin{aligned} m\left(\{(\mathtt{F},\mathtt{F}),(\mathtt{T},\mathtt{T})\}\right) &= 0.8\\ m\left(\Theta\right) &= 0.2\end{aligned}$$

The last, and perhaps the trickiest of the relations among the attributes is the relationship between R and M. If we have evidence that indicates that the ship breaks down on from 10 to 30% of the trips it makes after maintenance at the dock is performed, yet we are completely ignorant of what happens when we do no perform repairs, we get the following belief function over (R,M):

$$\begin{aligned} m_1\left(\{(\mathtt{T},\mathtt{T}),(\mathtt{T},\mathtt{F}),(\mathtt{F},\mathtt{F})\}\right) &= 0.1\\ m_1\left(\{(\mathtt{F},\mathtt{T}),(\mathtt{T},\mathtt{F}),(\mathtt{F},\mathtt{F})\}\right) &= 0.7\\ m_1\left(\Theta\right) &= 0.2\end{aligned}$$

If we have a separate body of evidence that indicates that the ship breaks down between 20 and 80% of the time after no maintenance, with no information about the cases when the ship was maintained, then we can construct the following belief function:

$$\begin{aligned} m_2\left(\{(\mathtt{T},\mathtt{F}),(\mathtt{T},\mathtt{T}),(\mathtt{F},\mathtt{T})\}\right) &= 0.2\\ m_2\left(\{(\mathtt{F},\mathtt{F}),(\mathtt{T},\mathtt{T}),(\mathtt{F},\mathtt{T})\}\right) &= 0.2\\ m_2\left(\Theta\right) &= 0.6\end{aligned}$$

We then combine these two belief functions using the direct sum operator. This is Smet's method of conditional embedding (see Shafer[1982]). It yeilds the following belief function:

$$\begin{aligned}
m(\{(\mathtt{T},\mathtt{T})(\mathtt{T},\mathtt{F})(\mathtt{F},\mathtt{F})\}) &= 0.06\\
m(\{(\mathtt{T},\mathtt{T})(\mathtt{T},\mathtt{F})\}) &= 0.02\\
m(\{(\mathtt{T},\mathtt{T})(\mathtt{F},\mathtt{F})\}) &= 0.02\\
m(\{(\mathtt{T},\mathtt{F})(\mathtt{F},\mathtt{T})(\mathtt{F},\mathtt{F})\}) &= 0.42\\
m(\{(\mathtt{T},\mathtt{F})(\mathtt{F},\mathtt{T})\}) &= 0.14\\
m(\{(\mathtt{F},\mathtt{T})(\mathtt{F},\mathtt{F})\}) &= 0.14\\
m(\{(\mathtt{T},\mathtt{T})(\mathtt{T},\mathtt{F})(\mathtt{F},\mathtt{T})\}) &= 0.04\\
m(\{(\mathtt{T},\mathtt{T})(\mathtt{F},\mathtt{T})(\mathtt{F},\mathtt{F})\}) &= 0.04\\
m(\Theta) &= 0.12
\end{aligned}$$

Finally, we come to the singleton belief functions, or prior belief functions. We can start by noticing that all of the singletons except F, L, and M will be vacuous. We now only have three which we need to define. They are specified in table III-2.

Attribute	$m(\{\mathtt{T}\})$	$m(\{\mathtt{F}\})$	$m(\Theta)$
L	0.3	0.5	0.2
F	0.2	0.6	0.2
M	0.0	1.0	0.0

Table III-2. Prior belief functions for Captain's Decision

We note that the belief function on M is a logical belief function. That is to say that it represents a rule that "we will not perform maintenance before this trip." It would probably be very valuable for the Captain to examine the sensitivity of the results to this rule. I will demonstrate that problem in a later paper.

Finally we note that if all of these belief functions are combined and we look at the margin on A, we get the belief function shown in table 1.

Bibliography

Buchanan and Shortlife[1984]. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA.

Dempster and Kong[1988]. "Uncertain Evidence and Artificial Analysis." Technical Report S-120, Harvard University, Department of Statistics. (To appear in the *Journal of Statistical Planning and Inference.*

Kong[1986a]. "Multivariate Belief Functions and Graphical Models." Ph.D. thesis, technical report S-107, Harvard University, Department of Statistics.

Kong[1986b]. "Construction of a Tree of Cliques from a Triangulated Graph." Technical report S-118, Harvard University, Department of Statistics.

Lauritzen and Spiegelhalter[1988]. "Fast Manipulation of Probabilities with Local Representations–With Applications to Expert Systems." *Journal of the Royal Statistical Society, Series B,*, Vol. 50.

Moussouris[1974]. "Gibbs and Markov Random Systems with Constraints." *Journal of Statistical Physics,* Vol. 10, pp 11-33.

Pearl[1982]. "Fusion, Propagation, and Structuring in Belief Networks." *Artificial Intelligence,* Vol. 29, pp 241-288.

Pearl[1986]. "Markov and Bayes Networks: A Comparison of Two Graphical Representations of Probabilistic Knowledge." Technical report R-46, University of California at Los Angles, Computer Science Department.

Rose, Tarjan and Lueker[1976]. "Algorithmic Aspects of Vertex Elimination on Graphs." *Siam J. Comput.,* Vol. 5, No. 2, pp 266-283.

Shafer[1976]. *A Mathematical Theory of Evidence.* Princeton University Press.

Shafer[1982]. "Belief Functions and Parametric Models." *Journal of the Royal Statistical Society, Series B,* Vol. 44, pp 322-352.

Shafer, Shenoy, and Mellouli[1986]. "Propagating Belief Functions in Qualitative Markov Trees." working paper no. 196, University of Kansas, School of Business.

Shenoy and Shafer[1986]. "Propagating Belief Functions with Local Computations." *IEEE Expert,* Vol. 1 No. 3, pp 43-52.

Tarjan and Yannakakis[1984]. "Simple Linear-Time Algorithms to test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs." *Siam J. Comput.,* Vol. 13, No. 3. pp 566-579.

Yannakakis[1981]. "Computing the Minimum Fill-In is NP-Complete." *Siam J. Alg. Disc. Meth.,* Vol. 2 No. 1, pp 77-79.