A Comparison of Architectures for Exact Inference in Evidential Networks

Boutheina Ben Yaghlane
LARODEC - ISG
IHEC Carthage, Tunisia
boutheina.yaghlane@ihec.rnu.tn

Abstract

This paper presents a comparison of two architectures for belief propagation in evidential networks, namely the binary join tree using joint belief functions [9] and the modified binary join tree using conditional belief functions [2]. This comparison is done from the perspective of graphical structure, message-passing scheme, computational efficiency, storage efficiency, and complexity analysis. As a main result, we show that the implication of the conditional relations between variables in evidential networks reduces the computational complexity of the inference process.

Keywords: Belief functions, belief propagation, evidential networks, binary join tree.

1 Introduction

One of the most important problems when using belief function theory [6] in practice is its relatively high computational complexity. The use of evidential networks (i.e. networks using belief functions) seems to be a promising solution of this problem. These networks are considered as a picture that provides an intuitive description of the problem. They are also considered as mathematical structures that specify different connections between the variables of a complex problem, transforming it into a clear representation. However, in evidential networks, the relations among the variables are usually represented by joint belief functions [7] rather than conditional belief functions. Nevertheless, the use of graphs to represent conditional independence relations is useful since an exponential number of conditional independence statements can be represented by a graph with a polynomial number of vertices [8].

In the belief reasoning literature, two architectures have been proposed for computing marginals of belief functions. The first one is the pioneering architecture for computing marginals, called the binary join tree (BJT) [9], which is an abstract framework for computing marginals applied to any domain satisfying some axioms. The second one is an adaptation of the BJT, called the modified binary join tree (MBJT) [2], showing how to represent independencies from the original directed evidential networks.

In this paper, we compare the BJT and MBJT architectures based on reasonable criteria determined during our experimentation. In particular, the graphical structure, the message-passing scheme, the computational efficiency, the storage efficiency, and the complexity analysis are discussed.

It is commonly believed that the BJT architecture is the most efficient architecture for computing marginals, but our experimental results show that the MBJT architecture is computationally more efficient than the BJT architecture and confirm the relevance of the use of conditional belief functions when inferring through evidential networks.

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 883–890
Torremolinos (Málaga), June 22–27, 2008

The paper is structured as follows. In Section 2, we first present the BJT architecture. Then, we introduce in Section 3 the MBJT architecture. Section 4 is devoted to a comparison between these two architectures on the basis of criteria specified progressively.

2 Binary Join Tree Architecture

The starting point is a directed evidential network of a given problem and some evidence (i.e. observations) for some variables. The task consists on computing the posterior belief for all variables in the network.

A *join tree* can be considered as a data structure that allows us to organize the computation. It consists in a set of nodes (corresponding to variables or subsets of variables of the problem) where each node is connected to one or more neighbor nodes and where the initially given potentials (or valuations) are distributed on the nodes. Marginals are then computed on the basis of a message-passing scheme, where nodes receive and send messages to their neighbor nodes.

Nevertheless, the join tree is not very efficient when multiple marginals have to be computed due to the redundant combinations. To get rid of this drawback, Shenoy introduces another data structure, called the *binary join tree* (BJT), which is a join tree according to which no node has more than three neighbors [9]. The basic idea behind a BJT is that all combinations are done in a binary basis.

In general, for a join tree, a node with m neighbor nodes would need m^2 combinations. However, each join tree can be transformed into a binary join tree through addition of other nodes involving the reduction of the number of combinations. Then, for a node with m neighbor nodes, $3(m-1)$ combinations are needed. Without the use of a binary join tree, a lot of unnecessary combinations would be performed for nodes which have many neighbor nodes.

The binary join tree is constructed by a process having as a main idea the *fusion algorithm*: a successive variable elimination. So when we delete a variable we combine all valuations that contain the variable in their domain, then we marginalize the variable out of the combination.

The propagation scheme through the BJT involves two-phase propagation: a *propagation up* and a *propagation down*. The arrival node for the *propagation up* is the departure node of the *propagation down* which is a root node. The formal procedure of the inference through the BJT is available in [9].

In the following, we present the graphical representation of the chest clinic problem [5] and the corresponding BJT.

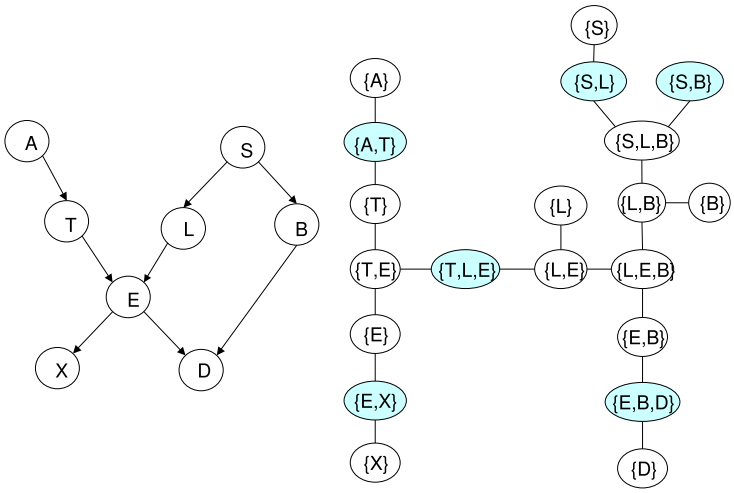

Figure 1: The directed evidential network and the corresponding binary join tree for the chest clinic problem.

3 Modified Binary Join Tree Architecture

It is important to realize that binary join trees represent a tradeoff between computing time and memory space. By storing intermediate results, less computing time is needed for the sake of using more memory space.

Special care has to be taken for nodes having a large number of neighbor nodes. For such nodes it happens that the combination of incoming messages generates huge mass func-

tions even if the incoming messages were relatively small. Nevertheless, it is still possible that every outgoing message is relatively small because marginalization was performed. In a binary join tree, these huge mass functions representing intermediate results would have been stored and therefore much memory space would have been lost.

Nevertheless, when we transform the original directed evidential network into a binary join tree, we lose some useful information about the relationships between the variables.

In order to avoid this drawback, Ben Yaghlane *et al.* [2] proposed a computational data structure based on the BJT and representing explicitly the (in)dependence relations of the original directed evidential network with conditional belief functions. This data structure is called the *modified binary join tree* (MBJT). Figure 2 gives the MBJT graphical structure of the chest clinic problem.

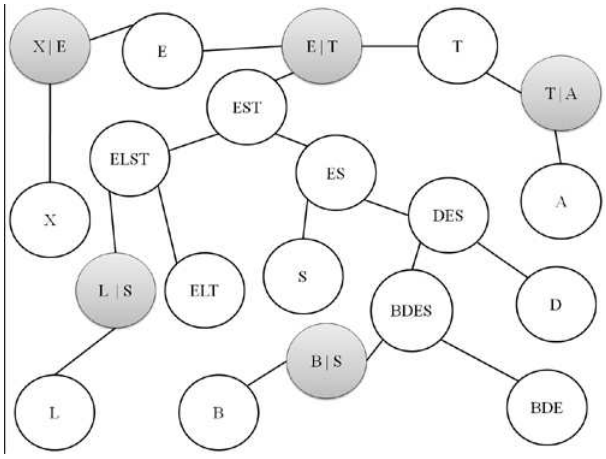

Figure 2: A MBJT tree for the chest clinic problem.

The belief propagation is then performed by applying two rules proposed by Smets [10], namely the *generalized Bayesian theorem* (GBT) and the *disjunctive rule of combination* (DRC). The full description of the propagation algorithms in the MBJT is given in [3].

4 Comparison Aspects

In this section, we discuss some aspects related to the algorithms used for BJT and MBJT architectures. In particular, we focus our attention to the graphical structure, the message passing scheme, the computational efficiencies, the storage efficiencies, and the complexity analysis of each architecture.

4.1 Graphical Structure

The BJT and MBJT architectures are two frameworks of nodes clustering. The first one takes as parameters an hypergraph and a set of valuations in order to construct the graphical model, initializes it and performs belief propagation. The result is a binary join tree [9], i.e. an undirected graph showing only undirected links between nodes. The second framework needs, as parameters, a directed evidential network weighted by conditional belief functions added to an hypergraph and a set of valuations. It also constructs the graphical model, initializes it and performs belief propagation so that to obtain a modified binary join tree, i.e. a mixed graph showing directed and undirected links between nodes.

When transforming graphical model, both frameworks adopt the idea of fusion algorithm. At the end of the construction of the first model, we obtain a BJT composed of nodes linked by joint belief functions. It doesn't show any (in)dependence relationship between the subsets of variables. Whereas, during the construction process, the MBJT profits from the structure of the original directed evidential network and takes advantage from its (in)dependence relations. As a result, we obtain a BJT composed of nodes linked by either conditional or joint belief functions. The (in)dependence relations between nodes in the MBJT are represented by the conditional nodes which are going to be useful when inferring through the MBJT.

In the BJT, nodes represent collections of variables belonging to subsets entered as parameters and several subsets of these cliques. In addition to these nodes, the MBJT is com-

posed of conditional nodes which are cliques of variables having conditional relations (children and parents) in the directed evidential network.

Since we include all singleton subsets during the construction of the graphical model, the graphical structure in both frameworks will yield marginals for singletons at the end of the message passing scheme.

4.2 Message Passing Scheme

Once the graphical model is constructed, the initialization process starts. During the BJT initialization, we only make use of joint belief functions, whereas the MBJT is initialized by means of joint belief functions for not conditional nodes (i.e. joint nodes) and conditional belief functions for conditional nodes.

In both frameworks, the propagation of messages weighted by beliefs is done in two phases: the *propagation-up* and the *propagation-down*. The root is determined according to the propagation-up scheme where messages are passing from leaves to the center of the tree.

On one hand, the inference in the BJT involves the totality of nodes created during the construction process. Every node in the BJT stores a potential and messages received from neighbors. So, every node is able to do computations.

On other hand, during the inference through the MBJT, only joint nodes are able to send and receive messages. For example, in the chest clinic problem, the MBJT (see figure 2) shows 20 nodes. Among these 20 nodes, we can find 5 conditional nodes, so only 15 remaining nodes will be involved in the inference process. This can reduce the amount of computations needed to propagate beliefs through the MBJT.

The other nodes which are conditional are not asked to receive, to send messages or to do computations because they are regarded as "bridges" between joint nodes. They determine whether the message sent from a joint node to another joint node is "parent" message or a "child" message. Therefore, they hold a decisive position allowing us to reveal who is the parent and who is the child before transferring messages between joint nodes.

In addition to the role of "bridges", these conditional nodes have another main role. They contribute to represent conditional (in)dependence between involved nodes. This confirms the capacity of the MBJT of representing conditional (in)dependence relationships between variable collections which offer more clearness to the graphical model and reveal explicitly the causal relations between subsets of variables.

For the MBJT, at the beginning we have potential representations of entered valuations, evidence representation and conditional belief functions representations which are deduced from the original directed evidential network. For the BJT, we have not conditional belief functions to represent at the beginning. At the end of belief propagation, we have marginals for all nodes in the BJT and marginals for only joint nodes in the MBJT.

4.3 Computational Efficiencies

As in the MBJT only joint nodes perform computations, there will be less computations in the MBJT than those in the BJT where all nodes must perform computations. In each node, computations are needed two times: first, when preparing messages to send to its neighbors, and second when computing marginals.

For transferring messages to its neighbors, each node has to perform m^2 combinations if it has m neighbors. In the worst case, the inference process involves n nodes and there will be $n*m^2$ combinations and $n*m$ marginalizations. In this case, the number of nodes determines the amount of computations when inferring through the graphical model. The MBJT involves less nodes than the BJT during the inference process because only joint nodes exchange messages between each other. We conclude then that the amount of computations in the MBJT is less extensive than the one in the BJT.

An execution of both algorithms (BJT and MBJT) under similar conditions and with the same parameters (entered subset, entered valuations, number of variables, heuristic, etc) confirms this conclusion. Notice that finding an optimal elimination sequence is very difficult. However, heuristics can be defined which produce nearly optimal elimination sequences. The basic idea is to look at each step only one step ahead (OSLA). Two such heuristics are known as

- OSLA Smallest Clique,
- OSLA Fewest Fill-ins,

respectively [1]. In order to perform programs, we fixed the parameters as follows:

- **Heuristic**: "$OSLA, SmallestClique$"
- **Elim. sequence**[1]: $ADTXBESL$
- **Observations** : No.

The program execution shows the following results (see Table 1) in which we use the following notations:

- **N**: the number of nodes
- **Comb**: the number of operations of combination
- **Marg**: the number of operations of marginalization
- **Msg**: the number of messages exchanged between nodes. For the MBJT architecture, **Msg** is represented as "total number of messages \ number of conditional messages"
- **Cond. nodes**: the number of conditional nodes in the MBJT

When analyzing briefly these results we deduce that the MBJT algorithm performs less operations than the BJT algorithm.

[1]Elimination sequence.

Table 1: Statistics after inference in BJT and MBJT architectures for the chest clinic problem

Architecture	Elim. sequence	N	Comb	Marg	Msg	Cond. nodes
BJT	ADTXBESL	20	92	58	38	
MBJT	ADTXBESL	20	72	31	16\12	5

4.4 Storage Efficiencies

In the BJT architecture, all nodes are storing joint belief functions, whereas in the MBJT architecture, there are two types of nodes: the joint nodes storing joint belief functions and the conditional nodes storing conditional belief functions.

In general storing a joint belief function allocates more memory space than storing a conditional belief function as shown in the following example.

Example 1. *Storing conditional belief function vs joint belief functions.*

Suppose that we have two variables X and Y defined on Θ_X and Θ_Y, respectively. To store a conditional potential $X|Y$ we need $|\Theta_Y| * 2^{|\Theta_X|}$. However to store a joint potential $X*Y$ we need $2^{|\Theta_Y|} * 2^{|\Theta_X|}$.

If $|\Theta_X| = 4$ and $|\Theta_Y| = 6$ then $X|Y$ needs $6 * 2^4 = 96$ memory units and $X * Y$ needs $2^6 * 2^4 = 2^{10} = 1024$ memory units.

Thus, we can appreciate the memory space gain when we store conditional belief functions instead of storing joint belief functions.

The potential storage memory space depends on the domain or the set of variables on which the potential is defined. For the case of the join tree, the domain is the set of variables composing the node because the potential is always defined on the domain of a node. In the following, the storage needed is described in terms of memory units.

The BJT is always storing joint belief functions, whereas the MBJT stores conditional belief functions and joint belief functions since it is composed of both conditional and joint nodes. This offers an important gain of memory space as shown in the following example:

Example 2. *Chest clinic problem storage.*

As mentioned in section 4.1, the BJT is composed of 20 nodes (only joint nodes) (see figure 1). Although, the MBJT is also composed of 20 nodes : 15 joint nodes and 5 conditional nodes (see figure 2). Suppose that every node is composed of 2 variables as an average node size. Suppose also that all variables have three states. To store all information about these 20 nodes,

- the BJT needs $20 * (2^3 * 2^3) = 20 * 64 =$ 1280 memory units
- the MBJT needs $(15 * (2^3 * 2^3)) + (5 * (3 * 2^3)) = 960 + 120 = 1080$ memory units

Notice that the above example is simple. The memory space gain is as important as the number of states of nodes is important.

In both architectures, the messages exchanged between nodes are stored in mailboxes. Indeed, every node has a mailbox in which it stores all messages exchanged by this node: the node receives messages from its neighbors in its mailbox and puts the messages to send in the mailboxes of the corresponding nodes. For the MBJT, only joint nodes have mailboxes for receiving messages from their neighbors. The conditional nodes store only conditional belief functions which will be useful when computing messages exchanged between two nodes on sides of the conditional nodes. Necessarily, this will reduce the complexity computational of the inference process.

4.5 Complexity Analysis

In practice, execution time of a program depends on the amount of data to deal with. There are programs or algorithms which behave better than other when the data set increases. For example, the execution time of an algorithm can increase exponentially relatively to the data set treated while another algorithm runs during a period increasing linearly to the data set dealt with.

Time is not always the best criterion to evaluate the performance of an algorithm. Indeed, the execution time does not prize exactly and faithfully the energy invested by the algorithm in performing a task because the time cost depends on the context of use (processor speed, available memory, process run in memory, etc). However the number of elementary operations does not depend on context of use and can be regarded as a criterion of performance evaluation. The number of elementary operations is relative to the data set to treat. Thus, we will speak about execution cost and not execution time.

Let S be the number of subsets of the initial $Sets$[2] entered as parameter when constructing the graphical model (MBJT or BJT), M be the maximum size of a subset, and p be the total number of nodes composing the graphical model (MBJT or BJT) obtained after performing the construction process. Only for the case where the graphical model is a MBJT, we consider n as the number of joint nodes and c the number of conditional nodes. Obviously $p = n + c$ since both algorithms generate frequently the same total number of nodes. We also need d as the size of the original directed evidential network.

4.5.1 Construction Algorithm

For the construction process, both algorithms have almost the same computational complexity, in other words the same amount of elementary operations. The construction step is relative to S and M, and more precisely to the product $M * S$.

When we observe the algorithms of construction for both architectures [4], we notice that they are considered as two imbricated loops and we can deduce that, in the worst case:

- **For the second loop** which is imbricated into the first loop, the algorithm iterates s times (s is the number of subsets truncated from the set $Sets$) because during every iteration the algorithm adds two selected subsets to the output set, performs the union of two subsets which is costing $(M + M)$ in the worst case,

[2]The set of subsets of variables for which we need marginals.

adds the subset resulted from the previous union and deletes both subsets; so adding one element and deleting two elements from $varsets$[3] until $|varsets| = 1$ will cost $2 * M * s$.

- **For the first loop**, in the worst case, the algorithm iterates S times, because during every iteration, the algorithm transfers two elements to $varsets$, deletes these two elements from $Sets$ and adds one element to $Sets$ until $|Sets| = 1$. Thus, the effort in first loop is relative to S.

Therefore, the computational complexity of the construction algorithm is relative to $S * (2 * M * s)$, in other words $O(S * M)$. This complexity analysis is valid for the algorithms of construction of both architectures. We notice that the MBJT construction algorithm undergoes, in addition to the cost of the classic construction algorithm, the effort of the algorithm for recognizing the conditional nodes having a computational complexity relative to $O(p * d)$.

4.5.2 Inference Algorithm

After constructing the graphical model, we obtain a set of nodes linked by edges. We will consider r as the maximum number of states for a variable and m the maximum number of variables in a node.

The inference algorithm in both architectures increases exponentially with the largest node size, so relative to $O(e^{2^{rm}})$. This computational complexity explains and justifies the use of heuristics, during the construction process, for minimizing the size of created nodes by determining the convenient variables elimination sequence in order to provoke the forming of nodes having the minimum possible node size.

Here, we can notice that the objective of these heuristics is compatible with the MBJT approach. Indeed, obtaining nodes with small size is leading to minimize the number of variables belonging to the node. This last aspect will allow the forming of the maximum of conditional nodes since a conditional node should have a small size because in the original directed evidential network the conditional relations connect in general a small (not large) number of variables. This provides more computational performance. We recall that a conditional node should contain, as variables, only those involved in conditional relations in the original directed evidential network.

The belief propagation through the BJT is relative to p, r and m whereas the belief propagation through the MBJT is relative to n, c, r and m.

- For the BJT: $O(p * 2^{rm})$ with $p = n + c$. Thus, $O((n * 2^{rm}) + (c * 2^{rm}))$
- For the MBJT: $O((n * 2^{rm}) + (c * r^{m/2} * 2^{rm/2}))$

To compare the complexity of both algorithms, we have to compare the second term $\{c * r^{m/2} * 2^{rm/2}\}$ and $\{c * 2^{rm}\}$. To show the difference between both architectures, let us take the following example:

Example 3. *Chest clinic problem computational complexity.*

The following values are assigned to the parameters: $r = 3$, $m = 4$, $p = 20$, $n = 16$, and $c = 4$.

So, the expression (for BJT): $\{c * 2^{rm}\}$ is equivalent to $4 * 2^{3*4} = 4 * 64 * 64 = 4 * 4096 = 16384$, and the expression (for MBJT): $\{c * r^{m/2} * 2^{rm/2}\}$ is equivalent to $4 * 3^2 * 2^{3*2} = 4 * 9 * 64 = 2304$.

With numeric values given in the above example, we notice the difference of order of magnitude between the complexity of both architectures (BJT and MBJT). With these small values, the order of magnitude is near to 1/5 which shows clearly that the complexity decreases significantly when adopting the MBJT architecture.

This is a very interesting aspect since the computational complexity is one of the most important difficulties in considering the belief function theory, especially when solving

[3] The set of subsets containing the current variable.

real-life problems. The gain on computational complexity is increasing when the parameters have values which are more and more extensive. This is the case of real-life problems which contain so many variables, so many nodes and also numerous conditional relations between variables. The conditional nodes in the MBJT are as numerous as the conditional relations in the original directed evidential network are available.

5 Conclusion and future works

In this paper, we have compared two exact inference algorithms, the BJT and the MBJT. This comparison was based on reasonable criteria.

The experimental results showed that the MBJT approach is more convenient than the BJT when using conditional belief functions in evidential networks. This reduces the computational complexity of the belief function theory which is the main difficulty faced when applying the theory to real-life problems.

When experimenting the inference algorithms on some complex problems, we noticed that the exact inference algorithm always inflicts a non negligible execution cost to perform corresponding tasks. This reveals that it is important to look at approximation methods [11]. In any case, an approximated numerical result which is close to the exact value may often be sufficient for a user.

Acknowledgements

I thank the referees for their helpful comments and suggestions.

References

[1] R.G. Almond, *Graphical Belief Modeling*, Chapman and Hall, 1995.

[2] B. Ben Yaghlane, P. Smets, and K. Mellouli, "Directed evidential networks with conditional belief functions", In *Proc. of ECSQARU-2003*, T.D. Nielsen, and N.L. Zhang (Eds), LNAI 2711, Springer-Verlag. pp.291–305, 2003.

[3] B. Ben Yaghlane, and K. Mellouli, "Belief functions propagation in directed evidential networks", In *Proc. of IPMU'2006*, Paris, France, pp.1451–1458, 2006.

[4] B. Ben Yaghlane, and K. Mellouli, "Inference in directed evidential networks based on the transferable belief model", *Int. J. of Approximate Reasoning* (to appear), 2008.

[5] S.L. Lauritzen, and D.J. Spiegelhalter, "Local computation with probabilities and graphical structures and their application to expert systems", *J. R. Statistical Society B*, vol. 50, pp.157–224, 1988.

[6] G. Shafer, *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976.

[7] G. Shafer, P.P. Shenoy, and K. Mellouli, "Propagating belief functions in qualitative Markov trees", *Int. J. of Approximate Reasoning*, vol.1, pp.349–400, 1987.

[8] P. P. Shenoy, "Valuation networks and conditional independence", In *Uncertainty in Artificial Intelligence*, D. Heckerman, and A. Mamdani (eds), Morgan Kaufmann, San Mateo, Ca, USA, UAI'93, pp.191-199, 1993.

[9] P. P. Shenoy, "Binary join trees for computing marginals in the Shenoy-Shafer architecture", *Int. J. of Approximate Reasoning*, vol.17, pp.239–263, 1997.

[10] P. Smets, "Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem", *Int. J. of Approximate Reasoning*, vol.8, pp.1–35, 1993.

[11] N. Wilson, "Algorithms for Dempster-Shafer theory", In *Handbook of Defeasible Reasoning and Uncertainty Management*, Vol.5: Algorithms, D.M. Gabbay and P. Smets, Kluwer Academic Publishers, pp.421–475, 2000.