A Review on Distinct Methods and Approaches to Perform Triangulation for Bayesian Networks

M. Julia Flores and José A. Gámez

Departamento de Sistemas Informáticos & SIMD - $i^3\mathcal{A}$
Universidad de Castilla-La Mancha
Campus Universitario s/n. Albacete. 02071
{julia,jgamez}@info-ab.uclm.es

Summary. Triangulation of a Bayesian network (BN) is somehow a necessary step in order to perform inference in a more efficient way, either if we use a secondary structure as the join tree (JT) or implicitly when we try to use other direct techniques on the network. If we focus on the first procedure, the goodness of the triangulation will affect on the simplicity of the join tree and therefore on a quicker and easier inference process.

The task of obtaining an optimal triangulation (in terms of producing the minimum number of triangulation links a.k.a. *fill-ins*) has been proved as an NP-hard problem. That is why many methods of distinct nature have been used with the purpose of getting as good as possible triangulations for any given network, especially important for big structures, that is, with a large number of variables and links.

In this chapter, we attempt to introduce the problem of triangulation, locating it in the compilation process and showing first its relevance for inference, and consequently for working with Bayesian networks. After this introduction, the most popular and used strategies to cope with the triangulation problem are reviewed, grouped into two main categories: heuristics and stochastic algorithms. Finally, another *family* of techniques could be understood as those based in decomposing the problem.

1 Introduction: the Compilation Process

If we consider that an expert system is composed of two main elements: *Knowlegde Base* (KB) + *Inference Engine* (IE), then a probabilistic expert system could be interpreted as a Bayesian network that models the particular problem (KB) and a secondary structure where inference is performed, normally an associated join tree[1] (IE).

[1] Also known as junction tree.

M.J. Flores and J.A. Gámez: *A Review on Distinct Methods and Approaches to Perform Triangulation for Bayesian Networks*, StudFuzz **213**, 127–152 (2007)
www.springerlink.com

A Bayesian network[38] is made up of two elements:

- The directed graph $G = (\mathcal{V}, E)$, where $\mathcal{V}$ are the variables/nodes in the graph and E the set of edges present in the graph from which dependencies and independencies between domain variables can be extracted.
- The probability distribution, which is usually stored in the form of tables, including for every variable X_i its probability $P(X_i|pa(X_i))$ or just the prior probability $P(X_i)$ if the node has no parents[2]. This is because of the **factorisation rule** that states:

$$P(X_1, \ldots, X_n) \quad = \quad \prod_{i=1}^{n} P(X_i|pa(X_i)) \tag{1}$$

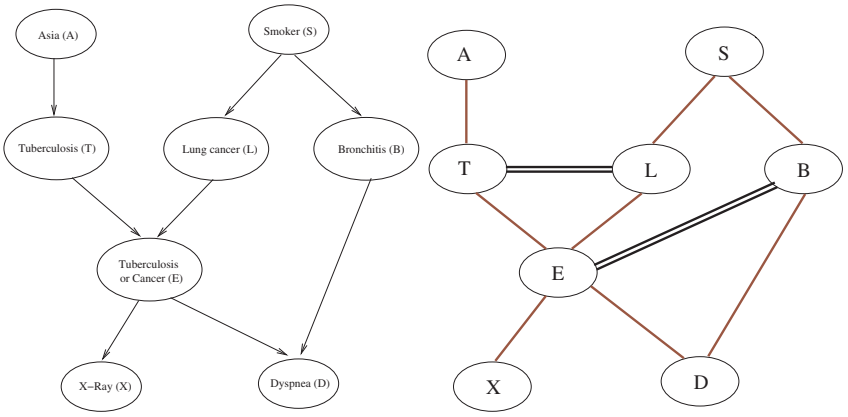
When inference is performed for Bayesian networks we normally want to obtain the posterior probability for the problem variables given some facts or evidence e that we have previously observed. Then, we wish to obtain the value of $P(X_i|e)$ either for all variables or for a subset of them. Computing this marginal and posterior probability is not always a simple task. The structure of the JT is a representation of the network, and being free of cycles, it allows certain algorithms to propagate probabilities that result in general quite more efficient. JT is also used for other inference tasks as the search of explanations or abductive inference [19; 37].

There exist efficient algorithms to propagate evidence in acyclic networks (polytrees) [38], but unfortunately this is not common to have networks under this constraint. What has been broadly used in order to propagate in an exact manner for any kind of network is some form of the so-called clustering method, which basically entails grouping in a single node (cluster) variables that are strongly related together. Then, these groups/clusters are organised as a tree and sophisticated adaptations to the evidence propagation algorithm can also be done such as Lauritzen & Spiegelhalter method [32], Shenoy & Shafer propagation [43] or Hugin architecture [26]. A more recent technique is Lazy propagation [34].

We can find many other methods based on the previous ones that seek a more efficient evidence propagation and whose main feature is the possibility of an approximate inference to improve even more the speed-up, for instance Penniless propagation [9]. There is even a combination of two different techniques as Lazy Propagation with Penniless [10].

As a means to obtain this tree of clusters the BN has to be *compiled*. Compilation is the process of transforming a Bayesian network into a secondary structure called junction tree and it is a step to preprocess the network in order to make inference in a more efficient way on the whole. However, the resulting join tree from the compilation process is not unique for a given BN. Thus, a quite interesting feature is to have the ability of choosing the best

[2] $pa(X_i)$ is the set of parent nodes for variable X_i, i.e. the nodes which have a link pointing to X_i.



(a) Graph. (b) Moral graph.

Fig. 1. Moralising graph for Asia network. Double lines indicate moral links.

tree among all possible ones for a particular BN. When using the join tree for inference, it is clear that the better this tree is the better our inference engine will work. Inference leads to a considerable number of operations, being our target to find a valid tree for our network, but also as simple as possible. We should note that a JT captures (in)dependence relations between variables. Nevertheless the groups of dependent variables might be bigger than necessary. Comparing two valid join trees, JT_1 and JT_2 related to the same network, the more complex the tree is, the more unnecessary dependencies it is actually including.

Let us indicate the four basic steps included in the compilation process[32]:

1. Obtain the *moral graph* from the original DAG that represented the BN.
2. Triangulate this *moral graph*.
3. Identify all the cliques.
4. Connect these cliques in order to form a valid join tree JT.

1.- Moralise the graph

The first step about moralisation takes the initial DAG G that forms the graphical part of the network and makes it undirected following these two rules: join those nodes with common parents by introducing a moral link[3], and drop directions of the directed edges. We show the graph for the *classical* network Asia or chest-clinic [32], originally directed (figure 1.(a)) and afterwards moralised (Figure 1.(b)).

[3] The origin of the term moral comes from *marrying* nodes with common children.

2.- Triangulate the moral graph

Second phase for compilation is the most problematic step: triangulation, since finding an optimal triangulation is an NP-hard problem [49]. To triangulate a graph it is needed to introduce a *chord* in those cycles of length greater than 3.

Normally, this process is done as the search of a deletion (or elimination) sequence σ which represents an ordering for all nodes in $\mathcal{V}$. Then, σ can also be seen as a function which relates every node $v_i \in \mathcal{V}$ with a unique number between 1 and $|\mathcal{V}|$. Therefore every node will have a position in the deletion sequence. Using this deletion sequence σ the necessary links to add (called *fill-ins*) will be obtained. If $adj(X_i)$ denotes the set of nodes adjacent to X_i in the undirected graph, then by *deleting* X_i we refer to the process of adding the necessary fill-ins in order to make $X_i \cup adj(X_i)$ a complete subgraph, and subsequently remove it and all its incident edges from the graph. The triangular graph G_T will be the result of adding to the moral graph the set ($\mathcal{F}$) of fill-ins added during the deletion process. That is, if $G_M = (\mathcal{V}, E_M)$ is the moral graph, then $G_T = (\mathcal{V}, E_M \cup \mathcal{F})$. Let us show one example for the moral graph of Asia network (Figure 2).

In Figures 2.(a)-(e) we proceed to use the deletion sequence σ showing the different steps for this ordering. So, as explained above, triangulation can be viewed as finding the deletion sequence. The method described in the previous paragraph is not complex, but the determination of a good deletion sequence is the most important step. For example, a sequence σ_2 as $\{D, S, L, B, E, T, A, X\}$ would produce the resulting triangulated graph shown in Figure 3.

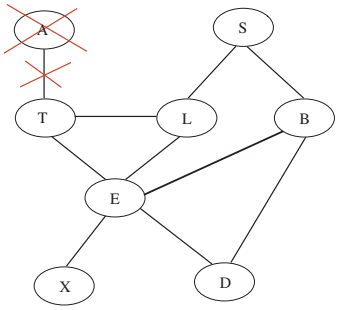
As we can see in Figure 3 the graph is correctly triangulated, since there are no cycles of length 4 or greater without a chord. However, we have introduced 4 fill-ins instead of the only one needed with σ sequence. That introduces more *unnecessary* relations among nodes that will make a more dense triangulated graph, and will construct bigger clusters. The size of a cluster is crucial for the efficiency of join tree-based algorithms. Notice that the triangulation could still introduce many more fill-ins, for example if variable E is the first to be removed, 8 fill-ins in only one step will be introduced!!. And Asia network is a very simple one, since it presents only 8 nodes. It is obvious that the number of possible sequences is equal to all the possible permutations ($|\mathcal{V}|!$), that is, it increases more than exponentially in the number of nodes.

3.- Identify the cliques

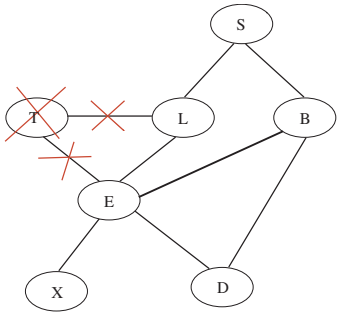
Once the graph is triangulated it is time to determine which are the cliques (clusters) in this triangulated graph. Now we can give a proper definition:

Definition 1. ***(Clique)*** Let G be an undirected graph, then all the maximal complete subgraphs in G are called cliques.

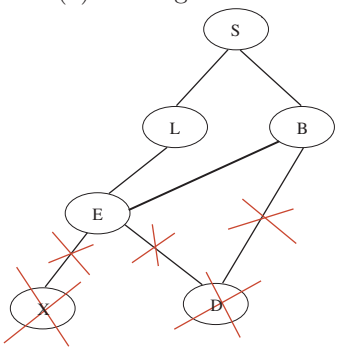
In our particular case we will be interested in identifying the cliques corresponding to the triangulated graph, G_T. As we have already explained, these



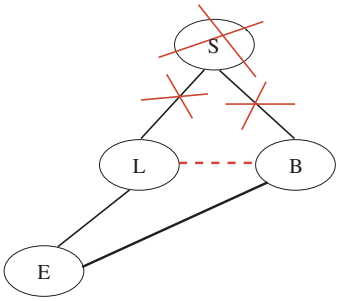
(a)Deleting node A



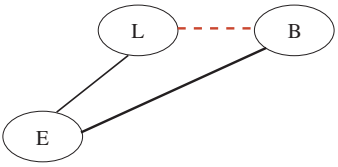
(b)When we delete T



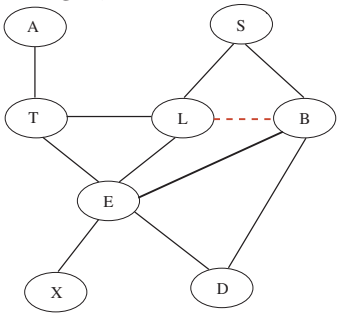
(c)Removal of X and then D



(d)Deleting S, which connects L and B



(e)And B, L, E (complete set)



(f)Triangulated graph: one fill-in = $\{(L, B)\}$

Fig. 2. Obtaining one possible triangulated graph for Asia. The used deletion sequence is $\sigma = \{A, T, X, D, S, B, L, E\}$.

cliques will be the nodes of the join tree. Since they are extracted from the triangulated graph they will also be dependent on the triangulation carried out, that is, on the introduced fill-ins.

Apart from determining the cliques we have to place them in a tree-shaped structure. And that leads us directly to the next step.

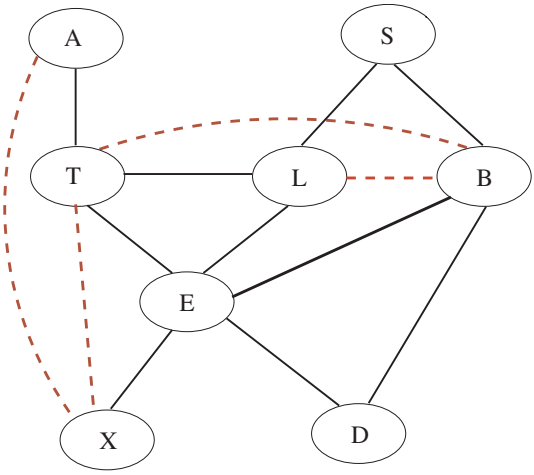


Fig. 3. Resulting triangulated graph when using deletion sequence $\sigma_2 = \{D, S, L, B, E, T, A, X\}$.

4.- Build the tree

It implies the establishment of the connections between cliques. From a triangulated graph there can be different possible join trees depending on the clique chosen as root, and sometimes a clique could be connected to different parents.

In order to guarantee that the running intersection property holds (see def. 2), we could use for example the maximum cardinality search [45] (MCS) with the aim of identifying the cliques and then connecting them in a tree.

Definition 2. ***Running Intersection Property:*** For every pair of clusters C_1 and C_2 whose intersection is not empty, that is, $V = C_1 \cap C_2 \neq \emptyset$, it is verified that V is contained in all nodes included in the path between C_1 and C_2.

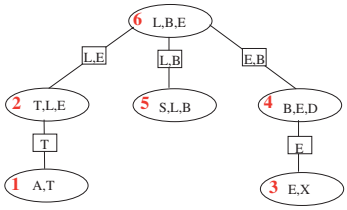
Apart from MCS there are other alternative methods of ordering the cliques if no deletion sequence is available. And, on the other hand, it is possible to construct the tree from a deletion sequence if we know the cliques formed when deleting v_i and taking the reverse order of these. Figure 4 shows this second procedure for the Asia example with the previous deletion sequence σ. In any case, all these methods use he same idea: when identifying the cliques we need to have them ordered in a certain way that will assure the running intersection property. So that, this order will lead to an iterative way of constructing the tree.

i	v_i	$Clique_i$
1	A	$\{A, T\}$
2	T	$\{T, L, E\}$
3	X	$\{E, X\}$
4	D	$\{E, B, D\}$
5	S	$\{S, L, B\}$
6	B	$\{L, B, E\}$
7	L	
8	E	

(a) σ order

i:	v_i	$Clique_i$	Sep_i	parents
8:	E	-		
7:	L	-		
6:	B	$\{L, B, E\}$	$\emptyset$	-
5:	S	$\{S, L, B\}$	$[L, B]$	**6**
4:	D	$\{B, E, D\}$	$[B, E]$	**6**
3:	X	$\{E, X\}$	$[E]$	6,**4**
2:	T	$\{T, L, E\}$	$[L, E]$	**6**
1:	A	$\{A, T\}$	$[T]$	**2**

(b) Inverse order



(c) Resulting join tree

Fig. 4. Ordering of the cliques from the triangulation $\sigma = \{A, T, X, D, S, B, L, E\}$ in Figure 2, identification of cliques and tree construction. Boldface when several options are possible indicates the randomly chosen one.

2 The Problem of Triangulation

As previously reviewed, the usual technique to triangulate a graph is selecting a *deletion/ elimination/ removing/ triangulation sequence* containing all the nodes in the graph. The method consists of an elimination process, following the sequence order, which will remove all the nodes. As pointed out before, finding an optimal deletion sequence is known to be an $\mathcal{NP}$-hard problem and coping with it involves a search over the space defined by all possible permutations of $|\mathcal{V}|$. Several approaches [40; 45; 28; 29; 8; 30; 24; 2; 1; 7; 20], most of them based on heuristics, have been proposed to search optimal solutions for this triangulation problem. Hence, these algorithms attempt to solve the problem of obtaining a good join tree from a BN, as next sections will show.

We should remark that these procedures to generate elimination sequences do not guarantee that we get an optimal triangulation either in terms of amount of added edges or in terms of the state space size when nodes are chosen randomly. Moreover, on average these measures in random sequences would normally be much larger than those corresponding to a minimum triangulation.

The question here is: *what we understand by an optimal triangulation?* If we refer to early work in graph triangulation what is understood by optimal triangulation is a *minimal triangulation*:

Definition 3. ***(Minimal Triangulation)***
If we have a triangulation $\mathcal{F}$ for an undirected graph G, $G_T = (V, E \cup \mathcal{F})$, denoting the set of fill-ins adding during triangulation, $\mathcal{F}$ is said to be **minimal** if $\nexists\ \mathcal{F}'$ so that $\mathcal{F}' \subset \mathcal{F}$ and $\mathcal{F}'$ is a valid triangulation for G.

That is, a triangulation $\mathcal{F}$ is minimal if for each fill-in $f \in \mathcal{F}$, the graph, $(\mathcal{V}, E \cup \mathcal{F} - \{f\})$ is not any more triangulated. Strongly related with this concept are those deletion sequences that constitute *perfect orderings*:

Definition 4. ***(Perfect ordering)***
Given an undirected graph $G = (\mathcal{V}$,E) and a sequence σ for $\mathcal{V}$, σ it is said to

be a *perfect ordering* if its use as deletion sequence to triangulate G does not produce any fill-in.

In fact, perfect orderings exist only for triangulated graphs and this is a way of checking if a given graph G is already triangulated.

In literature we can find many methods and studies for getting a minimal triangulation. The most known and first one is lexicographical search, LEX-M[40], providing a way of obtaining directly a minimal triangulation. The method consists of a particularly designed Breadth First Search (BFS), but labelling vertices (nodes) in a lexicographical way, LEX-BFS. LEX-M applies this labelling procedure along paths. More recent studies and (sometimes more efficient) methods have been designed [3; 6; 23; 39]. Among them, there is a recent successful technique [4] called MCS-M. This is a simplification of LEX-M where cardinality labels are used instead of lexicographical ones. In an analogical way, it applies the cardinality labelling of (neighbour) nodes along the path. MCS-M, as LEX-M, produces a minimal elimination ordering[4]. Even if both techniques could give different orderings it has been proved [47] that they create the same set of triangulations. LB-triang [5] is another recent algorithm that computes minimal triangulations with a computation complexity equal to the most efficient methods, and presenting certain properties that could make it especially interesting, such as it can also be implemented as an elimination scheme.

A different approach of obtaining a minimal triangulation is to follow an indirect path, that is, the method starts with a valid but non necessary minimal triangulation and then, it identifies the redundant fill-ins, so that eliminating them a minimal triangulation is obtained. If $\mathcal{F}$ is a set of fill-ins that make a graph G triangulated, $G = (\mathcal{V}, E \cup \mathcal{F})$, these methods identify a set of links $\mathcal{R}_{min} \subset \mathcal{F}$, so that $G_{min} = (\mathcal{V}, E \cup (\mathcal{F} \setminus \mathcal{R}_{min}))$ is minimally triangulated. The resulting minimal triangulation is therefore $\mathcal{F}_{min} = \mathcal{F} \setminus \mathcal{R}_{min}$. Among them, we find the method called *recursive thinning* designed by Kjærulf [28] and the algorithm proposed in [6].

The previous paragraphs discuss the problem of searching for minimal triangulations in the field of graph theory, however, when we move to the field of BNs triangulation things change. Now, the number of states of the variables plays a crucial role in the concept of optimal deletion sequences. Thus, in BNs triangulation a deletion sequence is optimal if it produces a triangulated graph whose associated join tree has minimal state space size. From this point of view a minimal triangulation does not need to be an optimal triangulation. As an example let us consider the Asia network previously introduced, where it is clear that triangulations $\mathcal{F}_1 = \{(L, B)\}$ and $\mathcal{F}_2 = \{(S, E)\}$ are minimal. If all the variables have two states (as it is the case) both triangulations are also optimal in the sense of state space size, but if we add an extra state to any of these variables, the two triangulations continue being minimal, but

[4] A deletion sequence that provides a minimal triangulation.

only one of them will be optimal with respect to the state space size criteria. In fact, Kjærulff [28] pointed out that most classical algorithms to look for a minimal triangulation are found to be highly ineffective when state space size is used as optimality criteria, being clearly surpassed by simple greedy heuristics (discussed in Section 3).

Although there is no general technique to perform always an optimal triangulation for any graph, there exist attempts to go as closer as possible as the algorithm QuickTree in [44], stated by the authors as the first algorithm that can optimally triangulate graphs with a hundred nodes in a reasonable time frame. In [21] we find a more modern *branch and bound* method, QuickBB with similar purposes. In [12] graph triangulation is interestingly stated and solved as a constraint satisfaction problem. Another an more recent example is found in the commercial tool Hugin[5] where one technique for optimal triangulation has been implemented. This particular method, as indicated in [25] is a combined exact/heuristic method capable of producing an optimal triangulation, but only if sufficient computational resources (primarily storage) are available.

Finally, as a remark, several research works have shown that all existing methods for local computation will imply (maybe in a *hidden* way) a triangulation task. Besides, those methods not using a secondary structure like the junction tree either are less efficient or present another problem of NP-hardness [27].

3 Heuristic Greedy Methods

This group of techniques is characterised by establishing an ordering criterion based on the search rule *"the next node to be deleted is that one minimising f()"* where $f()$ is in function of one or several measures over the set of nodes within the graph $G = (\mathcal{V}, E)$. The most used measures [28] are based on:

1. Nodes $i \in \mathcal{V}$:
   - Size.- the number of variables: $s(i) = 1$.
   - Weight.- logarithm of the natural size: $w(i) = log_2 c(i)$, where natural size, $c(i) = |\Omega_{X_i}|$, i.e. the number of states of variable X_i. Depending on the author *Weight* is seen directly as $c(i)$[6].
   - Incident.- number of incident links in node/variable i within the moral graph: $|adj(X_i)|$.
2. Groups or clusters $C_i \in \mathcal{P}(V)$:
   - Size of the group: $V(C_i) = \sum_{j \in C_i} s(j) = |C_i|$. Then it refers to the number of variables in the group (or clique).

[5] http://www.hugin.com

[6] And that will be the approach when *minWeight* is referred in this chapter.

- Weight: $W(C_i) = \sum_{j \in C_i} w(j)$.
  As it happened with the nodes, sometimes this name is used for denoting the natural size: $S(C_i) = \prod_{j \in C_i} c(j)$.
- (*Fill-ins*).- number of introduced edges while the triangulation process: $F(C_i)$. That is, the number of edges necessary to make the group complete except those links already belonging to the moral graph.

We should indicate that other authors use the term size also for the *weight* measure. In this work we will try to write clearly which criterion we are referring to.

From these enumerated measures a set of criteria appear that give rise (among other) to the following heuristics[7]:

• **Minimum size**.- This criterion is based on selecting as the next node to be deleted that one which minimises the function $f(C_i) = V(C_i)$. That is, at each step, it chooses the next variable, among those not yet deleted, which produces a clique of minimum size.

As Rose[42] noted minimum size heuristics is fast[8], but it presents some drawbacks:

- It does not produce, in general, a perfect ordering (see def. 4) if the graph is already triangulated.

- It does not generally produce minimal triangulations.

- There exist examples for which the produced triangulation is arbitrarily greater than the triangulation obtained by *minimum fill* (see below).

• **Minimum weight**.- This criterion is based on selecting as the next node to be deleted that one which minimises the function $f(C_i) = S(C_i)$. This heuristics presents exactly the same advantages and disadvantages as *minimum size*. Note that when all nodes have the same weight both heuristics are identical.

This heuristics gives good results on the whole. It tries to minimise the total sum of the cliques sizes by minimising, at each step, the size of every clique which is being created. This does not guarantee that the total tree space state size (or weight) is optimal, since choosing one variable that produces a minimal clique could force us to produce bigger cliques when other variables are deleted later. However, in general, this method provides trees which are relatively manageable.

In [8] another particular heuristics based on the same idea arise, but attempting to avoid its weak points. The main underlying idea of these heuristics is that in the moment of deleting a variable it should be sought to minimise

[7] we will assume that $C_i = \{X_i\} \cup adj(X_i)$

[8] It can be implemented in a computation time of order $O(|\mathcal{V}| + E')$, where $E' = E + |\mathcal{F}|$, being E the initial links and $\mathcal{F}$ those links added during triangulation.

the corresponding[9] $S(C_i)$. However, at the same time, the variable and all its corresponding links are deleted, which simplifies the resultant graph. Therefore, what they pursue is that this simplification for the resultant graph could also be taken into account.

Among the several heuristics that Cano and Moral [8] propose in their work, we find this approach called $H2$. This is very similar to *minimum weight*, at each case it chooses the variable X_i, among all the possible variables to be deleted, which minimises $S(i)/|\Omega(X_i)|$. With this feature when there are ties in (natural) size we remove first those variables of larger number of states, which leads to less complex cliques in the future formation of the tree.

• **Minimum fill**.- This criterion is based on selecting as the next node to be deleted that one which minimises the function $f(C_i) = F(C_i)$. In each case, it chooses the variable, among those not yet deleted, for which its elimination introduces a smaller number of fill-ins. This method presents the advantage of producing a perfect ordering when the graph is triangulated, but provokes the following drawbacks:

- It is slightly slower than the minimum weight heuristics, that is because the adjacency set for every node has to be explored regarding edges.

- In general it does not produce minimal triangulations.

There exist other heuristic techniques which attempt to tackle the problem of graph triangulation. In [24] they are classified in several groups:

1. Heuristics based on the relation between measures for nodes and clusters. They try to establish algebraic relationships between these two types of measures.
2. Heuristics based on measures for clusters and environments of nodes. They define the *k-neighbourhood* of a node by a distance k, which is determined as the minimum number of edges to go from one node to the other.
3. Compound heuristics. This sort of heuristics can be conceived as a hybridisation where the criterion to be used will vary on the different temporal stages of the triangulation process.
4. Iterative heuristics. Instead of using a single heuristic criterion to eliminate a node, they can make several iterations (each one with a different measure) in order to decide. They could be of k-iterations, where k could go from 1 (classical approach) until n ($n = |\mathcal{V}|$)). 2-iterations methods are studied in [24].

Since the complexity of finding a minimal triangulation grows as $n!$, it is not possible to carry out an exhaustive search directly, except when n is very small. Nevertheless, to construct an elimination order successively and to stop the execution when the total sum of the weights for the cliques (produced until this moment) exceeds the current smallest weight of a complete ordering could be of use to make an exhaustive search even for moderate-size graphs. Being

[9] Each deleted variable produces a group of variables, and when this is maximal it will therefore produce a clique.

triangulation an NP-complete problem, we can not generally expect that a *branch-and-bound* algorithm could find an optimal ordering within certain time limits. That is, the algorithm should finish either when the number of vertices exceeds a certain limit or when the number of the permutations left as discarded increases too slowly. Of course, the initial ordering will have a huge impact on the algorithm success. Thus, a *branch-and-bound* algorithm should be preferably used combining it with another quite faster algorithm (the first would be the last to apply) able of setting a "good" initial ordering for it with the goal of avoiding examining too many useless orderings and also with the goal of minimising the distance to some minimum ordering (we assume that low cost orderings are closer to a minimum one than a high cost ordering).

We could observe that the mentioned heuristics are only one-step lookahead, i.e., they just take into account that node which minimises a certain criterion if this node was deleted in the next step. We could then think of other heuristics able to look further than the next step. Unlike the heuristics above explained, about those looking beyond the next step, there is not much literature. This makes us think that, although they must produce better triangulations than the former, this improvement is not very significant in contrast to the complexity increase.

4 Methods Based on Stochastic Heuristics

The methods reviewed in section 3 present a good trade-off between the quality of the obtained deletion sequence (i.e. its associated join tree size) and the amount of computational resources (CPU time) required. Therefore, this kind of methods are suitable for *on-line* triangulation, that is, when there is a direct interaction between the user (knowledge engineer) and the compilation process and so a quick response is required. However, there are some occasions in which compilation can be carried out *off-line* and the time requirement can accordingly be relaxed. This is the case of compiling the final product (join tree or inference engine) to be given to the final user. At this stage, our goal should be to produce a junction tree as good as possible, because hundreds or thousands of propagations will be carried out over it. Thus, at this stage we can spend more time in the compilation process in order to achieve a better junction tree, and as a result algorithms requiring more CPU time are suitable.

When CPU time is not a strong constraint a family of algorithms arise as a good choice: *stochastic heuristic algorithms*. These algorithms are (in general) instances of metaheuristics that include stochastic behaviour so as to try to escape from local optima. Below we review some different approaches to the triangulation problem by using three outstanding representatives of this family of algorithms.

4.1 Simulated Annealing

Simulated annealing was (to our knowledge) the first stochastic heuristic used to solve the problem of Bayesian networks triangulation [29; 48].

Simulated annealing (SA) [50] is a stochastic optimisation algorithm used to look for global optima of NP-complete combinatorial problems having many local optima. SA is similar to a *hill climbing* algorithm, but sometimes it accepts to move to a worst solution in order to avoid to be trapped at local optima. The probability of accepting bad moves is controlled by a parameter t called *temperature*. Initially, during the exploration phase the temperature should be *high* in order to easily accept bad moves (exploration phase), but in successive iterations the temperature is decreased according to a cooling procedure and the probability of accepting cost-increases also decreases (exploitation phase).

When designing a SA algorithm for a given problem, different components have to be specified. Here we describe the algorithm proposed in [29].

- The search space is defined as all the possible elimination orderings for $\mathcal{V}$ (i.e., $|\mathcal{V}|!$ permutations).
- The neighbourhood of a deletion sequence σ is defined as all the deletion sequences $\{\sigma'\}$ obtained from σ by interchanging two of its nodes (positions).
- The cost/fitness of a deletion sequence is measured as the state space size of its associated join tree.

These three design decisions together with an appropriate cooling schedule are enough to have a SA algorithm that solves the triangulation problem, however Kjærulff [29] adds the following improvements in order to enhance the performance of the algorithm:

- Local computation of neighbour configurations (sequences). An efficient method is proposed that evaluates a new deletion sequence by only considering the cliques obtained when deleting the variables between the two interchanged positions.
- An additional parameter is introduced: the *radius*. The idea is to define a window w (length), such that, only positions inside this window can be interchanged. Initially a large window is set, so that free motion in the search space can be done (exploration). However, when the search process advances the window is reduced and so uniquely close neighbours are explored (exploitation). Because of the local evaluation proposed, this parameter is strongly related with the CPU time efficiency of the algorithm, since the smaller the window is the more efficiently the neighbours are evaluated.

The experiments carried out in [29] show that depending on the graph, on the average the state-space size of the join trees obtained by *minWeight* heuristics are 3 or 5.5 times larger than those obtained by SA.

4.2 Genetic Algorithms

SA carried out a local search that tries to escape from local optima by using a Monte Carlo method. On the contrary, Genetic Algorithms [22; 35] (GAs) do a *global* search by using a population of candidate solutions instead of a single one. In a GA we start with an initial population having solutions distributed over all the search space, then all the solutions are evaluated and a new population is obtained by: (1) selecting some of the individuals of the previous population (usually, higher fitness implies higher probability of being selected); (2) recombining some of the individuals of the previous population, that is, two individuals (parents) are selected and two off-spring are obtained by applying a crossover operator that mixes the representation of the parents; and (3) mutating some of the selected individuals (with low probability a small change is carried out over the individuals).

In the case of Bayesian networks triangulation the first contributions based on GAs are [24; 30]. The main features of the GA developed in [30] (that obtains similar results to the SA algorithm described in [29]) are:

- Individuals are represented by permutations (deletion sequences).
- The initial population is randomly generated in order to have initial points distributed uniformly through the search space.
- The selection mechanism is based on the rank of the individuals according to their fitness.
- A steady state GA is used. That is, in each generation instead of replacing the whole population, only a pair of off-spring are generated (selection+crossover+mutation) and (only if they are better) they replace the two worst individuals of the current population.
- Among the different specific operators for the case of permutations, Larrañaga et al. [30] found the combination of CX as crossover and ISM for mutation to be the best ones.

After these initial proposals different authors have used GAs to look for optimal deletion sequences. Concretely, Gámez and Puerta [20] slightly modified the algorithm proposed in [30] (where simplicial nodes are previously removed and informed initialisation of the population is used) obtaining better results (in terms of CPU time and join tree size).

4.3 Ant Colony Algorithms

The third stochastic heuristics we are going to review is *Ant Colony Optimisation* (ACO) [14].

Combinatorial optimisation based on *ant colony systems* is a recent metaheuristics that takes its basis on one aspect of ant behaviour, the ability to find shortest paths. Thus, in ACO a set of *artificial* ants (or agents) is used to look for the shortest paths in the same way as a *real* ant will do it: *following the pheromone track*. Concretely, when an artificial ant is located in a branch and

has to take a decision, it makes a probabilistic decision biased by the amount of pheromone deposited on the different branches. Due to the fact that the shortest paths are more frequently visited, they receive a higher amount of pheromone and thereby become more attractive for the subsequent ants. In this way the amount of pheromone plays the role of memoristic information, but in ACO the decisions are based as well on heuristic information. Thus, in the initial Ant System when an ant k is located at node i, it chooses node j as the following node to be visited with a probability proportional to:

$$p_k(i,j) = \begin{cases} \frac{[\tau_{ij}]^{\alpha} \cdot [\nu_{ij}]^{\beta}}{\sum_{u \in J_k(i)} [\tau_{iu}]^{\alpha} \cdot [\nu_{iu}]^{\beta}} & \text{if } j \in J_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where τ_{ij} is the amount of pheromone in edge (i,j); ν_{ij} represents *heuristic* information (knowledge) about the problem; $J_k(i)$ is the set of nodes for which there is a direct path from node i and not yet visited by ant k; and α and β are two parameters used to control the relative importance of pheromone with respect to heuristic information.

Although this is not a complete description of ACO-based algorithms (see e.g. [14]), one of the main differences in regard to previous discussed metaheuristics is already evident: in ACO algorithms is very easy to integrate problem domain knowledge. The use of heuristic knowledge in ACO algorithms helps to focus upon the search process (and speed it up), and this is just the point studied in [20] where ACO algorithms are applied to the Bayesian networks triangulation problem. Below, we describe the main points of the approach presented in [20]:

- *Representation.* The first thing we need is a graph over which ants will walk. In [20] the complete graph defined over the network variables is used in such a way that it is always possible to reach a node i from a node j for every pair of nodes (i,j). In consequence, there is a graph-form representation equivalent to the one used for the TSP (Travelling Salesman Problem [13]), but in the asymmetrical case, on account of the fact that it is not generally the same deleting X_i before deleting X_j as in the reverse order.
- *Reduction.* In this work simplicial nodes are removed before starting the combinatorial optimisation problem. In this way the search space is (in general) drastically reduced and the search results faster.
- *Heuristic knowledge.* In ACO algorithms the heuristic knowledge is usually static, that is, it can be calculated before any ant is launched. This is not the case in the triangulation problem, because the knowledge associated to edge (i,j) does not only depend on itself, but on the nodes previously visited (deleted). In [20] each ant implements a greedy heuristics (minSize, minWeight, ...), that is, each ant carries out a triangulation over its own copy of the graph. In this way, the matrix of pheromone will be a global structure, while the heuristic knowledge will be local to each ant.

- *Origin nodes.* As solutions are permutations any node will be valid as the starting point. However in triangulation it has no sense to consider (equally) all the nodes as possible origins. Thus, in [20] the probability of a node to be chosen as origin is calculated as a function of its heuristic value (minFill, minSize, ...).
- *Transition rule.* A variant of the rule described in eq. 2 is used in [20]:

$$j = \begin{cases} \arg\max_{u \in J_k(i)} \left\{ [\tau_{iu}] \cdot [\nu_{iu}]^{\beta} \right\} & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{3}$$

  where q is a random number uniformly distributed in [0,1], and $J \in J_k(i)$ is a node selected according to Eq. (2) with $\alpha = 1$. This is the rule proposed in Ant Colony Systems algorithms and it explicitly allows tuning (q_0) the amount of effort devoted to exploration/ exploitation.

Different experiments over a set of real and artificially generated networks are carried out in [20]. The obtained results turned out to be quite successful, regarding both accuracy and efficiency. Thus, ACO algorithms always obtain (on average) deletion sequences better that GAs, and due to the heuristic knowledge they use, the number of evaluated permutations (deletion sequences) is considerably smaller, making this approach faster than the one based on GAs. Furthermore, it presents also the advantage of having an ant-autonomy feature that could make them fit perfectly in a parallel environment with the aim of gaining efficiency.

5 Methods Based on Decomposition Techniques

Apart from the two previous approaches which are probably the most widely used, other triangulation techniques can be found in the literature, such as divide and conquer techniques based on the concept of *treewidth*[10] [1; 2]. The idea here is to use a different algorithm to triangulate in which the minimum vertex cut method is needed [15]. At each iteration it finds a minimum set of vertices X which being removed from graph G splits it into two disconnected components A and B such that $A \cup B \cup X = V$. This set X is then called the *minimum vertex cut*. This general algorithm proceeds in the two smaller problems $G[A \cup X]$ and $G[B \cup X]$, that is, those subgraphs obtained by projecting G on $A \cup X$ and $B \cup X$ respectively. And it goes on in this way so that each subgraph is triangulated such that X becomes a clique in it. As we will see MPSD is somehow based on this principle as well.

Within these techniques based on decomposition another related research line is the socalled *recursive hypergraph partitioning* (or simply hypergraph partitioning). They are quite broadly used in the context of VLSI design [41], but we can find some example of its application to join trees [11].

[10] Treewidth = number of variables, minus one, included in the biggest clique in the join tree

There exists another method capable of simplifying the triangulation task. In this case, it deals with a process to be performed prior to triangulate with the chosen method. In bibliography we can find it with different names, being *simplicial* (def. 5) the most broadly used. In [24] it is presented as *reduction*, and consists in eliminating all those nodes that, together with their neighbours, form a complete subgraph, i.e., no fill-in has to be added. This part of the network is then already triangulated and deleting them is not going to add any new fill-in. Another approach uses the application of preprocessing rules in order to reduce the graph [7]. In this approach the authors have developed a set of sophisticated *safe reduction rules* (being the first one removing simplicial nodes as well) to apply onto the graph before triangulation. The results are good, since a smaller (sub)graph has to be triangulated, but the technique requires more computation time than greedy heuristics.

Definition 5. ***(Simplicial node)***
Let $G = (\mathcal{V}, E)$ be an undirected graph. A node $N \in \mathcal{V}$ is said to be **simplicial** if this node N together with its set of neighbours,$\{N \cup adj(N)\}$, form a complete node set.

5.1 A Recent Triangulation Approach Based on the *Divide & Conquer* Methodology

In this section we are going to describe the method *triangulation by retriangulation* which combines some of the philosophies previously noted. Firstly, as treewidth-oriented techniques, it uses a method for dividing the total graph in smaller components. Olesen and Madsen[36] launched the possibility of applying the Maximal Prime Subgraph (MPS) Decomposition to the problem of triangulation. So, the idea is to retriangulate separately each MPS, since it has been proved to be perfectly valid for the final result. And, secondly, for those portions it will apply some methods of triangulation based on the procedures to get a elimination sequence reviewed above. Then, the work in [16] exploited the previous idea by using both greedy heuristic algorithms and stochastic ones (genetic algorithms).

5.2 Maximal Prime Subgraph Decomposition

It is clear that the decomposition of an undirected graph can be used as a tool for the triangulation procedure. We can consider the problem as a set of solvable subgraphs, following *divide and conquer* philosophy (see Fig. 5).

In this particular case the Decomposition using Maximal Prime Subgraphs (MPSD)[11] of an undirected graph constitute an intermediate step in a new approach for triangulation. This idea [36] consists of working separately on different parts of the initial graph. The triangulation for each graph will be

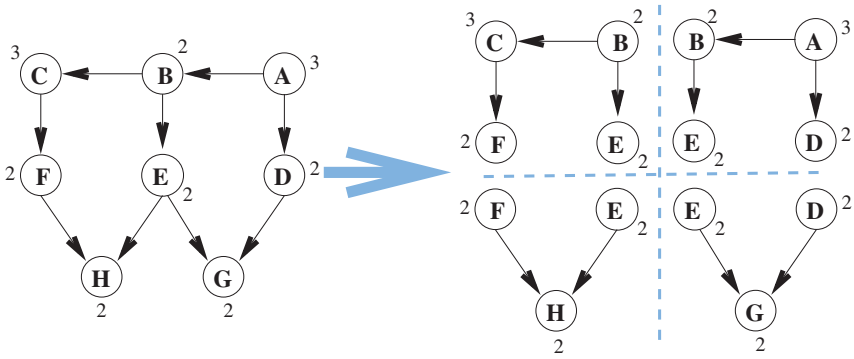


Fig. 5. Trying to reduce the problem of triangulating a network with n nodes to a set of k smaller subproblems: triangulate each subgraph S_k separately.

done separately and the global solution will then be the sum/combination of local solutions for smaller and independent graphs.

Let us just formalise the concept of maximal prime subgraph, for that, we also introduce the definition for decomposition (def. 6) of a graph and the characteristic for a graph of being decomposable (def. 7). Both of them can be easily related from previously presented ideas, since for constructing the JT we have made some kind of decomposition (MPS Tree will be the one which accomplishes the complete separators condition) whereas triangulated graphs are guaranteed to be decomposable [31] and that is somehow the justification for the necessity for a triangulation step. That is the reason why from here, we will refer to a decomposable graph as a triangulated graph.

Definition 6. ***(Graph decomposition)***
Let $G = (\mathcal{V}, E)$ be an undirected graph, and let A and B be two sets of vertices in G, G can be decomposed in A and B if and only if the following conditions are satisfied:

- $A \cup B = \mathcal{V}$,
- $A \setminus B \neq \emptyset$,
- $B \setminus A \neq \emptyset$,
- Both $A \setminus B$ and $B \setminus A$ are separated by $A \cap B$ and
- And $A \cap B$ is a complete subset (called clique separator).

Definition 7. ***(Decomposable graph)***
If a graph G and its subgraphs can be decomposed recursively until all the subgraphs are complete, then the graph is decomposable[12].

[11] Also known as decomposition by clique separators.
[12] Note that a graph can be decomposed without being decomposable.

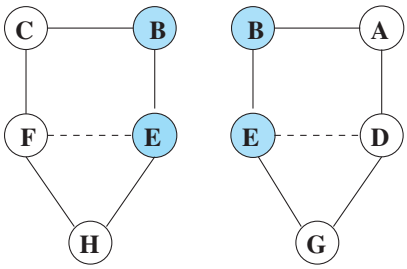


Fig. 6. A simple example of graph decomposition where $\{B, E\}$ is the clique separator for the BN in Fig. 8.a.

Then, it is said that a graph is *reducible* if it can be decomposed, that is, its set of nodes contains a clique separator, otherwise the graph is said to be *irreducible/**prime**/non-separable*. And this leads directly to def. 8:

Definition 8. ***(Maximal Prime Subgraph)***
A subgraph $G(A) = (\mathcal{V}, E)^{\downarrow A}$ of a graph G is a Maximal Prime Subgraph of G if $G(A)$ is irreducible and $G(B)$ is not irreducible $\forall B$ so that $A \subset B \subseteq \mathcal{V}$.

Finally, from the previous concepts it just remains to indicate what the Maximal Prime Subgraph Decomposition is[13]:

Definition 9. ***(Maximal Prime Subgraph Decomposition)***
Let $G = (\mathcal{V}, E)$ be an undirected graph. Its Maximal Prime Subgraph Decomposition is the set of induced maximal prime subgraphs of G resulting from a recursive decomposition of G.

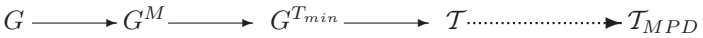


Fig. 7. Graphical process that indicates how to reach the MPST $\mathcal{T}_{MPD}$ from a Bayesian network $BN = (G, \mathcal{P})$, using as an intermediate step the join tree $\mathcal{T}$.

To obtain the MPSD of an undirected graph [45; 46; 33], the method in [36] is especially interesting for us, since it is based on the join tree constructed from a BN. The decomposition of the graph in MPSs is returned in a form of a tree: the Maximal Prime Subgraph Decomposition Tree (MPST), sometimes denoted as $\mathcal{T}_{MPD}$. Figure 7 shows graphically this process to obtain the MPST in an schematic way. The MPST will express by itself a decomposition (every tree node will denote a group of variables belonging to the same MPS). We could say basically that once the triangulation from which a join tree is

[13] It can be proved that this decomposition is unique for an undirected graph, as it is the moral graph.

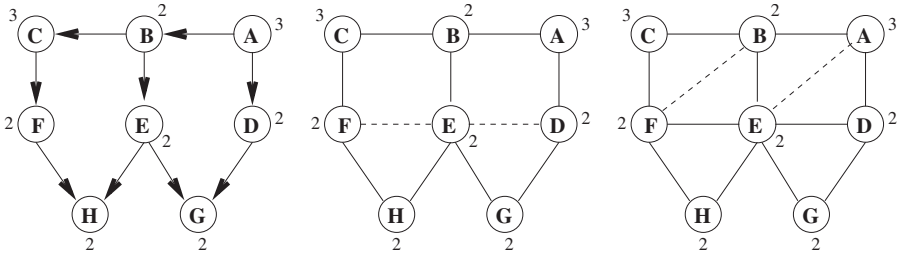


Fig. 8. Example of a Bayesian network (a), its associated moral graph (b) and a possible triangulation for it (c). Numbers next to each node indicate the number of states for the corresponding variable

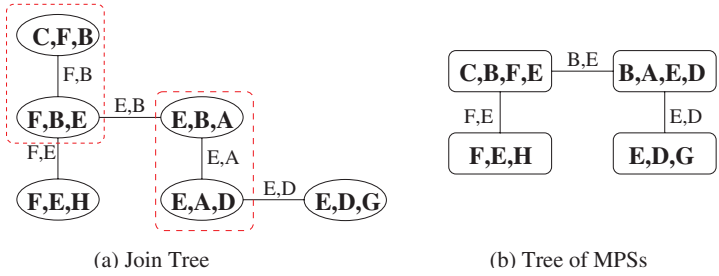


Fig. 9. Construction of the MPSs tree and the obtained result.

obtained is assured to be minimal, if we aggregate those cliques whose separators are not complete in G^M, we obtain the corresponding maximal prime subgraphs. If we have the network, moral graph and triangulated graph from Figure 8, then Figure 9 will show a corresponding join tree and the associated MPS tree. Since this is the necessary to guarantee that triangulations are minimal (def. 3), this can be achieved simply by using *recursive thinning*.

5.3 Triangulation of Bayesian networks by re-triangulation

It has been proved that is perfectly valid [18] to triangulate every subgraph in an independent way from the rest, and make a global triangulation of the graph by the combination of these *partial* triangulations. Retriangulating a graph can be worthy, even when the same triangulation method is used twice. That is to say, the same triangulation method is applied (first) when triangulating the moral graph, and (secondly) when triangulating each MPS separately.

The algorithm of ReTriangulation is as listed here:

1. Obtain moral graph G^M from BN.

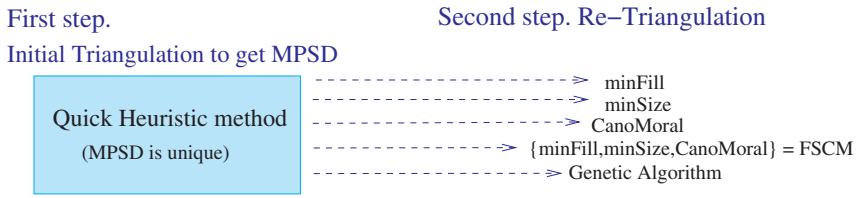


Fig. 10. Scheme for the retriangulation experiments: first step provides the decomposition and second step varies on the method M_i used to retriangulate the subgraphs S_i, giving place to the set of partial triangulations $\mathcal{F}_i$.

2. Using whichever triangulation method obtain the MPS decomposition $D = \{S_1, S_2, \ldots S_k\}$
3. Triangulate each S_i from D using a certain triangulation method M_i. Let $\mathcal{F}_i$ be the obtained triangulation for subgraph S_i.
4. Return the obtained result $\mathcal{F} = \cup_{i=1}^{k} \mathcal{F}_i$.

An experimental evaluation of this technique using it for real networks [16] was carried out, following the sketch in Figure 10. The obtained results have been quite satisfactory and should be regarded from two different points of view or goals:

1. Join Tree state space size:
   - Considering the heuristic techniques the tree size is generally better (smaller) when using the re-triangulation method, this difference is even bigger when we use a combined method called $FSCM$[14].
   - With respect to GA, the sizes of the obtained trees are quite similar.
2. CPU time:
   - Performing re-triangulation for heuristics implies a little more time, but this is due to the extra task of constructing the MPS Tree. This difference is only slightly noticeable for the heuristics because they are normally quicker.
   - FSCM is obviously three times slower than the rest (it tries the three methods), but since heuristic techniques are really quick and selecting the best one produces much better global results, it is a low price worth paying.
   - And the most important consequence related to time measuring is that a huge speed up is provided to GA. For example, in the case of Network Munin4 it can reduce (in this experiment settings) triangulation time from more than one day to less than 4 hours.

[14] It denotes a greedy technique that for every subgraph S_k tries the 3 different heuristics (minFill,minSize and CanoMoral) and it chooses the one that gives the best result, i.e., the smallest size. Since there is no an optimal heuristic for all cases, FSCM selects the best method M_i^* for every subgraph.

From these experiments and results we can mainly conclude that there exist some possibilities to optimise these results and to explore new combinations to get even better triangulations.

5.4 MPSD-based Incremental Compilation

In the last explained case, following the idea of *divide and conquer*, the natural decomposition of a graph into its prime subgraphs was exploited. In any manner, this decomposition tool is not reserved for triangulation itself, it can become even more powerful. The use of MPSD can be extended to the whole process of compilation. Since triangulation is the most *expensive* phase of compilation and this can be correctly and separately distributed among MPSs, we could sketch other techniques so that compilation could be less "dependent" on the global triangulation. For that, there exist a proposal to look more closely into the possibility of retriangulating some portions of the BN, and to use MPSD in order to perform incremental triangulations. This idea led directly to work on developing the approach of MPSD-Incremental Compilation of Bayesian networks [17]).

6 Main Conclusions

From this whole chapter and the analysed issues we can draw some main conclusions.

First, triangulation is still an unsolved problem, at least for a general case. There good techniques that might be *adjusted* depending on the problem, but not an optimal one (produced in a reasonable amount of time).

But on the other hand, triangulating has also been proved to be an unavoidable step in the computation of Bayesian networks. As a consequence, for solving queries and perform inference we must cope with this problem. This necessity for triangulating has brought about several endeavours to handle this problem, and the techniques found in literature are of distinct nature. So, we have shown most of the known approaches to tackle triangulation classifying them mainly in heuristic, stochastic algorithms and also techniques based on the division/decomposition of the problem. The last described algorithm (*ReTriangulation*) is of interest because it covers and integrates these three discussed manners of undertaking and solving the triangulation task.

Even though we find strong foundations for triangulation on the theory of graphs in literature, it is obvious that triangulation is still a quite open field to optimisation. It is illustrative to point out how this problem is already being studied in diverse mathematical and computing disciplines apart from Bayesian networks (probabilistic systems) such as the area of graph theory, VLSI (Very Large Scale Integration) circuits, data bases, constraint processing and graph algorithms.

References

[1] Amir, E. (2001) Efficient approximation for triangulation of minimum treewidth. In:*Proceedings of the 17th Conference on Uncertainty in Artificial Intelligene (UAI-01)*, 7–15

[2] Becker A. and Geiger D. (1996) A sufficiently fast algorithm for finding close to optimal junction trees. In: *Proceedings of the 12h Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, 81–89

[3] Berry A., Blair J.R.S. and Heggernes P. (2002) Maximum Cardinality Search for Computing Minimal Triangulations. *WG '02: Revised Papers from the 28th International Workshop on Graph-Theoretic Concepts in Computer Science.* In: Lecture Notes in Computer Science, **2573**, 1–12, Springer Verlag

[4] Berry A., Blair J.R.S., Heggernes P. and Peyton B.W. (2004) Maximum Cardinality Search for Computing Minimal Triangulations of Graphs. *Algorithmica* **39**(4): 287–298

[5] Berry A. , Bordat J-P., Heggernes P., Simonet G. and Villanger Y. A wide-range algorithm for minimal triangulation from an arbitrary ordering. To appear in *Journal of Algorithms*

[6] Blair J.R.S, Heggernes P. and Telle J.A. (2001) A practical algorithm for making filled graphs minimal. *Theoretical Computer Science* **205**(1-2):125–141

[7] Bodlaender H.L., Koster A.M. et al (2001) Pre-processing for triangulation of probabilistic networks. In: *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligene (UAI-01)*, 32–39

[8] Cano A. and Moral S. (1994) Heuristic algorithms for the triangulation of graphs. In: *Proceedings of the 5th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU)*, **1**, 166–171, Paris (France)

[9] Cano A., Moral S. and Salmerón A. (2000) Penniless propagation. *International Journal of Intelligent Systems*, **15**:1027–1059

[10] Cano A., Moral S. and Salmern A. (2002) Lazy evaluation in Penniless propagation over join trees. *Networks*, **39**:175–185

[11] Darwiche A. and Hopkins M. (2001) Using Recursive Decomposition to Construct Elimination Orders, Jointrees, and Dtrees. *Lecture Notes in Computer Science*, **2143**, 180–190, Springer Verlag.

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

[13] Dorigo M. and Gambardella L. (1997) Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, **1**:53–66

[14] Dorigo M. and Stützle T. *Ant colony optimization.* MIT Press, 2004.

[15] Even S. and Tarjan R.E. (1975) Network flow and testing graph connectivity. *SIAM Journal on Computing*, **4** :507–518

[16] Flores M.J. and Gámez J.A. (2003) Triangulation of Bayesian networks by retriangulation. *International Journal of Intelligent Systems* **18**(2):153–164

[17] Flores M.J., Gámez J.A. and Olesen K.G. (2003) Incremental Compilation of Bayesian networks. In: *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI–03)*, Morgan Kaufmann, 233–240

[18] Flores M.J. (2005) Bayesian networks inference: Advanced algorithms for triangulation and partial abduction. PhD thesis, Departamento de Sistemas Informaticos (Computing Systems Department), University of Castilla - La Mancha UCLM, Spain

[19] de Campos L.M., Gámez J.A., and Moral S. (2002) On the problem of performing exact partial abductive inference in Bayesian belief networks using junction trees. In: B. Bouchon-Meunier , J. Gutierrez, L. Magdalena, and R.R. Yager, editors, *Technologies for Constructing Intelligent Systems 2: Tools*, 289–302 Springer Verlag

[20] Gámez J.A. and Puerta J.M. (2002) Searching for the best elimination sequence in Bayesian networks by using ant colony optimization. *Pattern Recognition Letters.*, **23**:261–277

[21] Gogate V. and Dechter R. (2004) A Complete Anytime Algorithm for Treewidth. In: *Proceedings of the Twentieth Annual Conference on Uncertainty in Artificial Intelligence (UAI–04)*, pp. 201–208, AUAI Press

[22] Goldberg D.E. (1989) *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley

[23] Heggernes P. and Villanger Y. (2002) Efficient Implementation of a Minimal Triangulation Algorithm. In: *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms.* Lecture Notes in Computer Science, **2461**, 550–561, Springer Verlag

[24] Hernández L.D. (1995) Diseño y validación de nuevos algoritmos para el tratamiento de grafos de dependencias (Validation and design of new algorithms to dependency graph processing.) Doctoral thesis, Dpto. de Ciencias de la Computación e I.A. Universidad de Granada, Spain

[25] HUGIN Expert A/S. *API manual for the Hugin Decision Engine V6.3.* http://developer.hugin.com/documentation/API_Manuals/

[26] Jensen F.V., Lauritzen S.L. and Olesen K.G. (1990) Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, **4**:269–282

[27] Jensen F.V. and Jensen F. (1994) Optimal junction trees. In: *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, 360–366, Morgan-Kaufmann

[28] Kjærulff U. (1990) Triangulation of graphs - algorithms giving small total space. Technical Report R 90-09, Department of Mathematics and Computer Science. Institute of Electronic Systems. Aalborg University

[29] Kjærulff U. (1992) Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, **2**:7–17

[30] Larrañaga P., Kuijpers C.M., Poza M. and Murga R.H. (1997) Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, **7**:19–34

[31] Lauritzen S.L., Speed T.P. and Vijayan K. (1984) Decomposable graphs and hypergraphs. *Journal of the Australian Mathematical Society* Series A **36**: 12–29

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

[33] Leimer H.G. (1993) Optimal decomposition by clique separators. *Discrete Mathematics*, **113**:99-123

[34] Madsen A.L. and Jensen F.V. (1999) Lazy Propagation: A Junction Tree Inference Algorithm based on Lazy Evaluation. *Artificial Intelligence*, **113** (1-2):203–245, Elsevier Science Publishers, North-Holland

[35] Michalewicz Z. (1996) *Genectic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag

[36] Olesen K.G. and Madsen A.L. (2002) Maximal prime subgraph decomposition of bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics*, Part **B**:(32), 21–31

[37] Park J.D. and Darwiche A. (2003): Solving MAP Exactly using Systematic Search. In: *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI–03)*, Morgan Kaufmann, 459–468

[38] Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo

[39] Peyton B.W. (2001) Minimal Orderings Revisited. *SIAM Journal on Matrix Analysis and Applications*, **23**(1):271–294

[40] Rose D., Tarjan R.E. and Lueker G.S. (1976) Algorithmic aspects of vertex elimination graphs. *SIAM Journal on Computing*, **5**:266–283

[41] Selvakkumaran N. and Karypis G. Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization *IEEE Transactions on Computer Aided Design* (in press)

[42] Rose D. (1972) A graph theoretic study of the numerical solution of sparse positive definite systems of linear equations. In: R. Reed ed. *Graph Theory and Computing*, 183–217, Academic Press, New York

[43] Shafer G.R. and Shenoy P.P. (1990) Axioms for probability and belief-function propagation. In: R.D. Shachter, T.S. Levitt, L.N. Kanal and J.F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, **4**, 169–198, Elsevier Science Publishers B.V. (North-Holland)

[44] Shoikhet K. and Geiger D. (1997) Practical algorithm for finding optimal triangulations. In: *Proceedings of the National Conference on Artificial Intelligence*, AAAI, USA, 185–190

[45] Tarjan R.E. and Yannakakis M. (1984) Simple linear-time algorithms to test chordality of graps, text acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, **13**:566–579

[46] Tarjan R.E. (1985) Decomposition by clique separators. *Discrete Mathematics*, **55**:221–232

[47] Villanger Y. (2004) LEX M versus MCS-M. Technical Report Reports in Informatics 261, University of Bergen, Norway

[48] Wen, W. X. (1990) Decomposing belief networks by simulated annealing. In: (C. P. Tsang, ed.) Proceedings of the Australian Joint Conference on Artificial Intelligence, 103–118

[49] Wen, W.X. (1991) Optimal decomposition of belief networks. In: P.P. Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, **6**, 209–224 North-Holland

[50] Kirkpatrick S., Gelatt C.D. and Vecchi M.P. (1983) Optimization by simulated annealing. *Science*, **220**:671–680