Heuristic Algorithms for the Triangulation of Graphs*

Andrés Cano and Serafín Moral

Departamento de Ciencias de la Computación e I.A.
Universidad de Granada
18071 - Granada - Spain

Abstract. Different uncertainty propagation algorithms in graphical structures can be viewed as a particular case of propagation in a joint tree, which can be obtained from different triangulations of the original graph. The complexity of the resulting propagation algorithms depends on the size of the resulting triangulated graph. The problem of obtaining an optimum graph triangulation is known to be NP-complete. Thus approximate algorithms which find a *good* triangulation in *reasonable* time are of particular interest. This work describes and compares several heuristic algorithms developed for this purpose.

1 Introduction

A number of different algorithms for the propagation of uncertainty in graphical structures have been developed in recent years. The main effort has been devoted to the study of probabilistic propagation. Original algorithms were proposed on a directed acyclic graph without loops by Kim and Pearl [12, 17]. Different propagation algorithms have been described for the general case [13, 18, 19, 6, 21, 15, 16].

Sachter, Andersen and Szolovits [20] have shown that the different exact algorithms described in the literature are all particular cases of a single general algorithm, called the clustering algorithm. This algorithm shows that the essence of the efficiency achieved in the original propagation algorithms, comes from the factorization of the global probability distribution which is obtained from the independence relationships among the variables of the problem. This factorization gives rise to the cluster tree, where all the computations are carried out. If we define the size of the cluster tree like the sum of the sizes of the cliques then the efficiency of the computations depends on the size of the associated cluster tree; in other words, efficiency is a polynomial function of size.

* This work was supported by the Commission of the European Communities under project DRUMS2, BRA 6156

The key point in the construction of the cluster tree is the triangulation of the undirected graph expressing the independences of the problem. Through this process, the most efficient exact algorithms can be obtained by considering the optimum triangulation (the triangulation with a minimum size of the associated clusters). However, the construction of the optimal triangulation is known to be an NP-complete problem [1].

Kjærulff [9] has studied different heuristic methods and a simulated annealing algorithm in order to obtain efficient triangulations within a reasonable amount of time. One heuristic was shown to be the best, with similar results to the simulated annealing algorithm, which is much more time consuming.

In this work we propose new heuristics and compare them with Kjærulff's best heuristic, by using different randomly generated graphs. It is shown that these heuristics can improve the optimality of the resulting cluster trees.

The obtained results are applicable to other uncertainty formalisms. Shafer and Shenoy [21] and Cano, Delgado Moral [2] have shown that propagation algorithms can be applied in order to propagate other uncertainties represented by other methodologies in which combination and marginalization are defined as elementary operations, verifying a set of axioms. The efficiency of the calculations will be also much related to the size of the associated cluster tree.

This work is organized in the following way: in the second section, we briefly give the fundamental concepts related to the problem of triangulation of a graph and the construction of the associated cluster tree. In the third section, we describe the different heuristics, that will be compared and evaluated in the fourth section. Finally, in the fifth section, we present the conclusions.

2 Graph Triangulation

Let us assume an n-dimensional variable $(X_1, \ldots, X_n)$, each variable, X_i taking values on a finite set U_i. A usual way of expressing independences among variables is by means of a directed acyclic graph in which each node represents a variable, X_i (see Fig. 2) [16].

From a directed acyclic graph we can build its associated undirected graph, also called the moral graph. This graph is constructed by adding undirected links between two parents of the same node and ignoring the direction of the links in the directed graph (see Fig. 2) [13].

An undirected graph is said to be chordal or triangulated if, and only if, every cycle of length four or more has an arc between a pair of nonadjacent nodes. To build a tree of clusters in which to carry out the computations we need a triangulated undirected graph. Graphs in Fig. 2 and 3 are triangulated and non triangulated graphs, respectively.

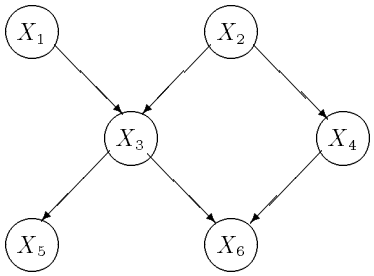

Fig. 1. A Directed Acyclic Graph

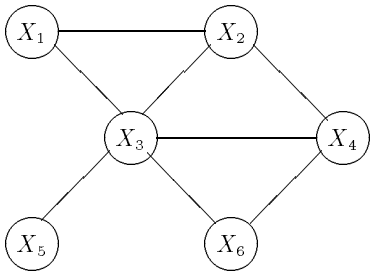

Fig. 2. The Moral Graph

A triangulated graph can be obtained from a general undirected graph by adding links. Different triangulations are obtained from different permutations of the nodes, $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$. Given a permutation, σ the associated triangulation is built according to the following procedure.

- For i=1 to n
  - Add links between the pairs of nodes adjacent to $X_{\sigma(i)}$. Let L_i the set of added links.
  - Remove node $X_{\sigma(i)}$ and all the links connecting $X_{\sigma(i)}$ with other nodes of the graph.
- To obtain the triangulated graph from the original graph, add all the links in the sets L_i $(i = 1, \ldots, n)$.

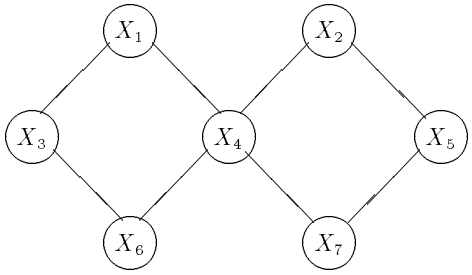

Fig. 3. A non chordal undirected graph

The permutation σ is called a deletion sequence. If we consider the deletion sequence $(\sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7)) = (3, 2, 7, 4, 1, 5, 6)$ in the graph in Fig. 3, we obtain the triangulated graph in Fig. 4.

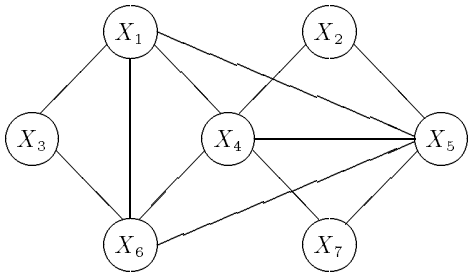

Fig. 4. Associated triangulated graph under σ

The clusters of a triangulated graph are cliques: maximal complete (all the nodes are adjacent) subgraphs. In the graph in Fig. 4 the cliques are $\{X_1, X_3, X_6\}$, $\{X_1, X_4, X_5, X_6\}$, $\{X_2, X_4, X_5\}$, $\{X_4, X_5, X_7\}$.

The size of a clique $\{X_i\}_{i \in I}$ is the number of elements of the cartesian product of the sets in which each X_i takes its values. That is the number of elements of $\Pi_{i \in I} U_i$, which is equal to the product of the number of elements of each U_i.

From a triangulated graph we can construct the cluster tree, also called the junction tree [14]. This can be done by the maximum cardinality search algorithm [23]. The graph in Fig. 5 is one of the possible tree of clusters associated to the triangulated graph in Fig. 4.

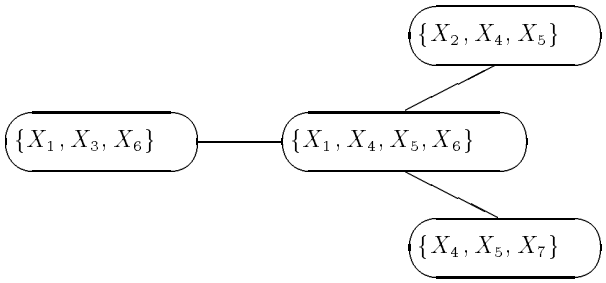

Fig. 5. The tree of cliques

The size of a tree of cliques is the sum of the sizes of each one of its cliques. To obtain efficient computations of this tree it is essential that its size be as small as possible.

3 Heuristic Algorithms

Olmsted [10], Kong [11], Kjærulff [9] proposed the following heuristic in order to obtain a good deletion sequence:

H1 *In each case, select the variable among the set of non-deleted variables producing a clique of minimal size and then delete this variable.*

This heuristic produces exceptional results in the general case. It attempts to minimize the sum of the sizes of the cliques by minimizing, in each step, the size of each of the cliques that are being created. This does not guarantee that the size of the tree of cliques is optimal; selecting a variable producing a minimal clique can force us to produce bigger cliques when deleting the other variables, but in general it produces relatively manageable trees.

The main idea underlying the new heuristics introduced in this work, is that when we delete a variable we are creating a clique with a size that should be minimized. However, at the same time, we are removing this variable and all the corresponding links, thereby simplifying the resulting graph. The simplification obtained in the resulting graph can also be a hepful guide in order to obtain efficient triangulations.

Following this idea we introduce the next heuristic algorithms to find a deletion sequence:

If X_i is a possible variable to be deleted then,

- $S(i)$ will be the size of the clique created by deleting this variable.
- $E(i)$ will be the number of elements of U_i.
- $M(i)$ will be the maximum size of the cliques of the subgraph given by X_i and its adjacents nodes.
- $C(i)$ will be the sum of the size cliques of the subgraph given by X_i and its adjacent nodes.

The heuristic algorithms follow the following rules,

H2 *In each case select a variable, X_i, among the set of possible variables to be deleted with minimal $S(i)/E(i)$.* This heuristic is similar to H1, but H2 computes the size of the environment of X_i (size of the clique to be produced deleting X_i) only with the adjacent nodes of X_i. In this way, we not only delete the variables with a less complex environment, but also make it possible to delete a variable with a big U_i.

H3 *In each case select a variable, X_i, among the set of possible variables to be deleted with minimal $S(i)-M(i)$.* $M(i)$ is the size of the biggest clique where X_i is included. After deleting X_i the biggest clique where X_i is included will have the size $S(i)$. With H3 we try to minimize the increment in the size of the biggest clique where each variable is included.

H4 *In each case select a variable, X_i, among the set of possible variables to be deleted with minimal $S(i) - C(i)$.* This heuristic is similar to H3, but the difference is computed with $C(i)$, that is the sum of the sizes of the cliques where X_i is included.

H5 *In each case select a variable, X_i, among the set of possible variables to be deleted with minimal $S(i)/M(i)$.*

H6 *In each case select a variable, X_i, among the set of possible variables to be deleted with minimal $S(i)/C(i)$.*

When we have two or more equally good variables to delete the next, we choose randomly one of that variables. We think that it would be possible to find a better tie-breaking.

The heuristics H5 and H6 are similar to H3 and H4 respectively, but rather than computing a difference, we compute the factor in the increment of the cliques. The complexity of H2 is equal to the complexity of H1. The idea of simplying the resulting graph is only partially considered by H2: the cause of deleting a variable X_i is the size of the created clique, $S(i)$. It is considered that the resulting graph is simpler if we delete a variable with more cases. By dividing $S(i)$ by the number of cases, $E(i)$, we balance the cost of deleting a variable with the simplicity of the obtained graph. However, $E(i)$ is not a very precise indicator of the simplicity of the graph; more precise indicators of this simplicity are considered in the heuristics H3, H4, H5 and H6. The intuitive basis of this rules are the following: if we delete variable X_i, the arcs of the old graph, and not present in the new graph, are the arcs linking X_i with its adjacents nodes. $C(i)$ and $M(i)$ aim to measure the complexity of the subgraph given by these arcs. But these rules are more complex than H1 and H2 because we must compute the cliques of a graph, although this graph will generally be small because it is the graph given by a node and its adjacent nodes.

4 Evaluation of the Heuristic Algorithms

We have implemented the different heuristic algorithms in C language in a SUN Workstation. To evaluate them, we have generated 500 graphs of each one of the following groups, and we are triangulated each one of the graphs with the six heuristics.

a) The graphs have 50 nodes which are are enumerated from 1 to 50. Each node has 2 cases, a number of parents chosen from the set $\{0, \ldots, 5\}$, and generated according to an uniform distribution of mean 2.5, rounding to the closest integer. The parents are the nodes immediately preceding the given node (this produces chain-like graphs).

b) The same as a), but now the parents are selected randomly among the preceding nodes. The nearest nodes have a higher probability of being chosen as parents.

c) The same as a), but now the parents are selected randomly among all of the preceding nodes. All the nodes have the same probability of being chosen as parents.
d) The same as a), but now the number of cases for each node is selected according to a Poisson distribution of mean 2 with a minimum of 2.
e) The same as b), but now the number of cases for each node is selected according to a Poisson distribution of mean 2 with a minimum of 2.
f) The same as c), but now the number of cases for each node is selected according to a Poisson distribution of mean 2 with a minimum of 2.
g) The same as a), but now the number of cases for each node is selected according to a Poisson distribution of mean 3 with a minimum of 2.
h) The same as b), but now the number of cases for each node is selected according to a Poisson distribution of mean 3 with a minimum of 2.
i) The same as c), but now the number of cases for each node is selected according to a Poisson distribution of mean 3 with a minimum of 2.
j) The same as a), but now the number of cases for each node is selected according to a Poisson distribution of mean 4 with a minimum of 2.
k) The same as b), but now the number of cases for each node is selected according to a Poisson distribution of mean 4 with a minimum of 2.
l) Same as c), but now the number of cases for each node is selected according to a Poisson distribution of mean 4 with a minimum of 2.

The mean size and standard deviation of sizes of the tree of cliques obtained by using each of the heuristic algorithms are given in tables 1 and 2.

Method	H1	H2	H3	H4	H5	H6
a)	489	489	428	483	428	483
b)	19.965	19.965	18.027	17.564	20.153	15.613
c)	23.378	23.378	22.826	22.660	26.046	22.152
d)	2.700	2.613	2.278	2.301	2.278	2.301
e)	2.158.184	1.606.454	2.199.538	2.091.678	1.691.858	1.052.726
f)	4.457.325	2.420.105	2.890.733	2.872.668	1.820.238	1.423.942
g)	6.606	6.407	5.548	5.558	5.548	5.558
h)	8.377.588	7.981.114	7.943.450	8.002.658	6.290.989	5.197.635
i)	65.147.965	32.137.124	53.160.398	53.028.113	28.502.411	24.144.396
j)	16.825	16.275	14.189	14.196	14.189	14.196
k)	130.974.850	128.772.625	131.136.043	133.913.541	123.910.850	77.734.851
l)	1.627.641.198	1.506.177.237	1.605.723.127	1.606.452.599	528.890.374	475.232.064

Table 1. Mean sizes of the tree of cliques

The results are the following:

- The new algorithms, with the exceptions of H3 and H4, are generally better than the previous known best algorithm, H1.
- The relative improvement increases as a function of the complexity of graphs. The mean size applying H1 divided by the mean size applying H6 is close

Method	H1	H2	H3	H4	H5	H6
a)	64	64	54	71	54	64
b)	34.332	34.332	27.561	28.929	30.047	23.179
c)	39.452	39.452	38.911	31.731	39.188	32.630
d)	1.097	1.063	956	954	956	954
e)	7.423.401	5.080.951	8.543.176	8.367.031	7.670.390	3.276.971
f)	50.441.890	13.401.449	18.944.797	18.946.028	7.086.978	5.472.380
g)	2.630	2.546	2.188	2.187	2.188	2.187
h)	21.490.256	23.676.980	21.069.018	21.380.543	19.737.130	16.726.049
i)	705.106.339	288.762.073	466.932.003	466.839.764	201.002.001	209.095.702
j)	7.940	7.636	6.916	6.914	6.916	6.914
k)	344.446.921	451.214.399	360.803.452	380.197.254	500.892.991	570.893.175
l)	25.081.485.346	24.760.156.861	24.797.026.066	24.796.998.549	4.180.926.330	4.280.997.524

Table 2. Standard Deviations of the sizes of the tree of cliques

to one in the case of the simplest graphs (type a). In the case of the more complicated graphs (type l) it is greater than 3.

- The best heuristic is H6. This heuristic has a greater computer cost than H1. However, H2 has the same cost as H1 and produces, in general, better results than H1 (sometimes it produces a factor of 2).
- When the parents of a node are chosen among its preceding nodes (cases a, d, g, and j) the graphs are relatively simple and all of the heuristic algorithms produce similar results.
- When the nearest nodes have a higher probability of being chosen as parents, (b, e, h, k), the graphs have a smaller size than when the nodes have the same probability (c, f, i, l), but we obtain larger sizes if we compare with the sizes of the simple cases (a, d, g, j).

5 Conclusions

In this work, we have presented new heuristic algorithms to triangulate a graph. The problem of triangulation of graphs is the key point for the efficiency of propagation algorithms in graphical structures.

We have presented new heuristic algorithms which have been tested with regard to that proposed by Kjærulff [9]. For this we have used 500 randomly generated graphs for each one of 12 different types of graphs. We have obtained better triangulations than with Kjærulff's heuristic.

Since the problem of obtaining an optimal triangulation is NP-hard, some new heuristics could be considered. More complex procedures could introduce further improvements, as in the case of simulated annealing algorithms proposed in [9].

The main problem is what amount of calculation should be devoted to the triangulation. That is, should we choose very complex triangulation procedures, giving rise to good trees of cliques, or very fast triangulation procedures, giving

rise to poor triangulations?. In general, the answer will depend on the particular problem we are trying to solve, but in general, we can say that the good trees of cliques can save time for a lot of different inference problems: the same tree will be used for several propagation cases, so, in most of the situations it will be worthy to spend some reasonable time on the triangulation.

The results of this work are useful for not only probabilistic propagation. The size of the associated tree of cliques is also fundamental for the propagation of uncertainty represented by other formalisms, such as belief functions and convex sets of probabilities.

References

1. Arnborg S., D.G. Corneil, A. Proskurowski (1987) Complexity of finding embeddings in a k-tree. *SIAM Jour. Alg. Discr. Meth.* **8**, 277-284.
2. Cano J.E., M. Delgado, S. Moral (1993) An axiomatic framework for the propagation of uncertainty in directed acyclic graphs. *International Journal of Approximate reasoning* **8** 253-280.
3. Chin H.L., G.F. Cooper (1989) Bayesian network inference using simulation. In: *Uncertainty in Artificial Intelligence, 3* (Kanal, Levitt, Lemmer, eds.) North-Holland, 129-147.
4. Cooper G.F. (1988) Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Stanford University, Stanford, California.
5. Cooper G.F. The computational complexity of probabilistic inference using bayesian belief networks is NP-hard. *Artificial Intelligence* **42**, 393-405.
6. D'Ambrosio B. (1991) Symbolic probabilistic inference in belief nets. Department of Computer Science, Oregon State University.
7. Geman S., D. Geman (1984) Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721-741.
8. Henrion M. (1986) Propagating uncertainty by logic sampling in Bayes' networks. Technical Report, Department of Engineering and Public Policy, Carnegie-Mellon University.
9. Kjærulff U. (1990) Triangulation of graphs-algorithms giving total state space. R 90-09, Department of Mathematics and Computer Science, Institute for Electronic Systems, Aalborg University.
10. Olmsted, S.M.(1983). On representing and solving decision problems. Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.
11. Kong, A. (1986). Multivariate belief functions and graphical models. Ph.D. dissertation, Department of Statistics, Harvard University, Cambridge, MA.
12. Kim J.H., J. Pearl (1983) A computational model for causal and diagnostic reasoning in inference engines, *Proceedings 8th IJCAI*, Karlsruhe, Germany.
13. Lauritzen S.L., D.J. Spiegelharter (1988) Local computation with probabilities on graphical structures and their application to expert systems. J. of the Royal Statistical Society, B 50, 157-224.
14. Jensen F. V. Junction trees and decomposable hypergraphs, *Research report*, Judex Datasystemer A/S, Aalborg, Denmark.

15. Pearl J. (1986) A constraint-propagation approach to probabilistic reasoning. In: *Uncertainty in Artificial Intelligence* (L.N. Kanal, J.F. Lemmer, eds.) North-Holland, 357-370.
16. Pearl J. (1986) Fusion, propagation and structuring in belief networks. *Artificial Intelligence* **29** 241-288.
17. Pearl J. (1988) *Probabilistic Reasoning in Intelligent Systems.* Morgan & Kaufman, San Mateo.
18. Shachter R.D. (1986) Evaluating influence diagrams. *Operations Research* **34**, 871-882.
19. Shachter R.D. (1988) Probabilistic inference and influence diagrams. *Operations Research* **36**, 589-605.
20. Shachter R.D., S.K. Andersen, P. Szlovits (1991) The equivalence of exact methods for probabilistic inference on belief networks. Submitted to *Artificial Intelligence.*
21. Shafer G., P.P. Shenoy (1990) Probability Propagation. *Annals of Mathematical and Artificial Intelligence* **2**, 327-351.
22. Shenoy P.P., G. Shafer (1990) Axioms for probability and belief-functions propagation. In: *Uncertainty in Artificial Intelligence, 4* (R.D. Shachter, T.S. Levitt, L.N. Kanal, J.F. Lemmer, eds.) North-Holland, Amsterdam, 169-198.
23. Tarjan R.E., M. Yannakakis (1984) Simple linear-time algorithm to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing* **13** 566-579.