An Objective function for Belief Net Triangulation

Marina Meilă and Michael I. Jordan
mmp@ai.mit.edu
Center for Biological & Computational Learning
Massachusetts Institute of Technology
79 Amherst Street E10-237A
Cambridge, MA 02139

Abstract

This paper presents a new approach to the triangulation of belief networks.[1] Triangulation is a combinatorial optimization problem; our idea is to embed its discrete domain into a continuous domain Θ. Then, by suitably extending the objective function over Θ, we can make use of continuous optimization techniques to do the minimization. We used an upper bound of the total junction tree weight as the cost function. The appropriateness of this choice is discussed and explored by simulations.

1 Introduction. What is triangulation ?

Triangulation is a basic step in the process of transforming a directed belief network into a *junction tree*. This process is known as *decomposition* and it consists of the following stages: first, the directed graph is transformed into an undirected graph by an operation called *moralization*. Second, the moralized graph is triangulated. A graph is called *triangulated* if any cycle of length > 3 has a *chord* (i.e. an edge connecting two nonconsecutive vertices). If a graph is not triangulated it is always possible to add new edges so that the resulting graph is triangulated. We shall call this procedure *triangulation* and the added edges the *fill-in*. In the final stage, the junction tree [6] is constructed from the *maximal cliques*[2] of the triangulated graph. We define the state space of a clique to be the cartesian product of the state spaces of the variables associated to the vertices in the clique and we call *weight* of the clique the size of this

[1]Part of this work is also presented in [7]

[2]A *clique* is a fully connected set of vertices and a maximal clique is a clique that is not contained in any other clique.

state space. The *weight of the junction tree* is the sum of the weights of its component cliques. All further exact inference in the belief net takes place in the junction tree representation and the number of computations required by an inference operation is proportional to the weight of the tree.

For each graph there are several and usually a large number of possible triangulations, with widely varying state space sizes. Moreover, triangulation is the only stage where the cost of inference can be influenced. It is therefore critical that the triangulation procedure produces a graph that is optimal or at least "good" in this respect.

Unfortunately, this is a hard problem. No optimal triangulation algorithm is known to date. However, a non-optimal triangulation is readily obtained; a simple algorithm is Rose's *elimination procedure* [8] which chooses a node v of the graph, connects all its neighbors to form a clique, then eliminates v and the edges incident to it and proceeds recursively. The resulting filled-in graph is triangulated.

It can be proven that the optimal triangulation can always be obtained by applying Rose's elimination procedure with an appropriate ordering of the nodes. It follows then that searching for an optimal triangulation can be cast as a search in the space of all node permutations. The idea of the present work is the following: embed the discrete search space of permutations of n objects (where n is the number of vertices) into the continuous set of doubly stochastic matrices of dimension n. This set is a simplex whose extreme points are matrices representing permutations. By suitably extending the cost function to the continuous domain we have transformed the discrete optimization problem into a continuous nonlinear optimization task. This allows us to take advantage of the thesaurus of optimization methods that exist for continuous cost functions.

This idea is developed in section 3. Section 2 introduces the cost function that we used, which is an upper bound on the junction tree weight that is easier to compute over our domain. The same section also discusses the relationship to other objective functions for triangulation. Section 4 evaluates both the cost function and our method by simulations. Section 5 contains final remarks.

2 The objective

In this section we introduce the objective function that we used and we discuss its relationship to the tree weight. We also review other possible choices of cost functions and the previous work that is based on them.

First we introduce some notation. Let $G = (V, E)$ be a graph, its vertex set and its edge set respectively. Denote by n the cardinality of the vertex set V, by r_v the number of values of the (discrete) variable associated to vertex $v \in V$, by $\#$ the elimination ordering of the nodes, such that $\#v = i$ means that node v is the i-th node to be eliminated according to ordering $\#$, by $\mathrm{n}(v)$ the set of neighbors of $v \in V$ just before its elimination (thus possibly including other neighbors) ===and by $C_v = \{v\} \cup \{u \in \mathrm{n}(v) \mid \#u > \#v\}$.[3] Then, a result in [4] allows us to express the total weight of the junction tree obtained with elimination ordering $\#$ as

$$J_{(\#)} = \sum_{v \in V} \mathrm{ismax}(C_v) \prod_{u \in C_v} r_u \qquad (1)$$

[3]Both n and C_v depend on $\#$ but we chose not to emphasize this in the notation for the sake of readability.

where ismax(C_v) is a variable which is 1 when C_v is a maximal clique and 0 otherwise. As stated, this is the objective of interest for belief net triangulation. Any reference to optimality henceforth will be made with respect to J.

This result implies that there are no more than n maximal cliques in a junction tree and provides a method to enumerate them. This suggests defining a cost function that we call the *raw weight* J' as the sum over all the cliques C_v (thus possibly including some non-maximal cliques):

$$J'_{(\#)} = \sum_{v \in V} \prod_{u \in C_v} r_u \tag{2}$$

J' is the cost function that will be used throughout this paper.

Another objective function, used (more or less explicitly) by [9] is the size J^F of the fill-in:

$$J^F_{(\#)} = |F_{\#}| \tag{3}$$

where $F_{\#}$ is the set of edges added by the elimination algorithm. There exists a method, the *lexicographic search* [9], that finds minimal triangulations with respect to J^F, but finding the minimum one is NP-hard [10]. It can be proven [8] that for a constant number of values r_v per node, the minimal triangulations with respect to J^F are also local minima for J and J'. Even if most of the local minima found by lexicographic search were good enough (something that is not supported by practical experience [5]), the problem with this algorithm is that it takes into account only topological information, ignoring the values of r_v. As our simulation will show, this is an important drawback.

Kjaerulff introduced the *minimum weight* heuristic [5], a greedy minimization method for J' (that overcomes the aforementioned problem) and later a simulated annealing approach [6] that explicitly optimizes J.

Becker [3] introduced recently a triangulation algorithm which is not based on node elimination. The algorithm minimizes the *cliquewidth* J^C, which is the largest clique-weight in the junction tree.

$$J^C_{(\#)} = \max_v \prod_{u \in C_v} r_u. \tag{4}$$

J^C is coarser than J in the sense that the same J^C can correspond to permutations with different values of J. But we expect that with the increase of r_v and of the graph density the cost of the largest clique will tend to dominate J improving the agreement between the two criteria. Optimizing J^C is provably NP-hard [1].

Now back to J'. A reason to use instead it of J in our algorithm is that the former is easier to compute and to approximate. But it is natural to ask how well do the two agree?

Obviously, J' is an upper bound for J. Moreover, it can be proved that if $r = \min r_v$

$$J_{(\#)} \leq J'_{(\#)} < J_{(\#)} \frac{r}{r-1}\left(1 - \frac{1}{r^{|V|} - 1}\right) \tag{5}$$

and therefore that J' is less than a fraction $1/(r-1)$ away from J. The bound is attained when the triangulated graph is fully connected and all r_v are equal.

In other words, the differece between J' and J is largest for the highest cost triangulation. We also expect this difference to be low for the low cost triangulation. An intuitive argument for this is that good triangulations are associated with a large number of smaller cliques rather than with a few large ones. But the former situation means that there will be only a small number of small size non-maximal cliques to contribute to the difference $J' - J$, and therefore that th agreement with J is usually closer than (5) implies. The simulations in section 4 will further support our choice.

3 The continuous optimization problem

This section shows how to define J' over the continuous domain of doubly stochastic matrices.

A *doubly stochastic matrix* θ is a matrix for which the elements in a row or column sum to one.

$$\sum_i \theta_{ij} = \sum_j \theta_{ij} = 1 \quad \theta_{ij} \geq 0 \quad \text{for } i, j = 1, ..n. \tag{6}$$

When θ_{ij} are either 0 or 1, implying that there is exactly one nonzero element in each row or column, the matrix is called a *permutation matrix*. Permutations will be denoted in the forthcoming by $\#$. $\theta_{ij} = 1$ and $\#i = j$ will both mean that the position of object i is j in the given permutation. The set of doubly stochastic matrices Θ is a convex polytope of dimension $(n-1)^2$ whose extreme points are the permutation matrices [2] . Thus, every doubly stochastic matrix can be represented as a convex combination of permutation matrices.

Let us define new variables μ_{uv} and e_{uv}, $u, v = 1, .., n$. For any permutation $\#$

$$\mu_{uv} = \begin{cases} 1 & \text{if } \#u \leq \#v \\ 0 & \text{otherwise} \end{cases} \qquad e_{uv} = \begin{cases} 1 & \text{if the edge } (u,v) \in E \cup F_{\#} \\ 0 & \text{otherwise} \end{cases}$$

where E is the set of edges and $F_{\#}$ is the fill-in.

In other words, μ represent precedence relationships and e represent the edges between the n vertices. With these variables, J' can be expressed as

$$J'_{(\theta)} = \sum_{v \in V} \prod_{u \in V} r_u^{\mu_{vu} e_{vu}} \tag{7}$$

For a matrix θ in the interior of the simplex, μ and e will take values in $[0,1]$ but the form of J' shall stay the same. We have defined over Θ as

$$\begin{aligned} \mu_{uv} &= 1 - \mu_{vu} = \frac{1}{1 - \sum_i \theta_{ui}\theta_{vi}} \sum_{j<i} \theta_{uj}\theta_{vi} \quad u, v, i, j = 1, ..n \quad \text{and } v \neq u \\ \mu_{vv} &= 1 \end{aligned}$$

To define e we use a result in [9] stating that an edge (u, v) is contained in $F_{\#}$ iff there is a path in E between u and v containing only nodes a for which $\#a < \min(\#u, \#v)$.

$$e_{uv} = e_{vu} = \begin{cases} 1 & \text{for } (u,v) \in E \text{ or } u = v \\ \max\limits_{P \in \{paths\ u \to v\}} \prod_{a \in P} \mu_{au}\mu_{av} & \text{otherwise} \end{cases}$$

The above assignments give the correct values for μ and e for any set of θ values representing a permutation. Over the interior of the domain, e is a continuous, piecewise differentiable function. Each e_{uv}, $(u, v) \notin E$ can be computed by a shortest path algorithm between u and v, with the length of $(a, b) \in E$ defined as $(-\log \mu_{au}\mu_{bv})$.

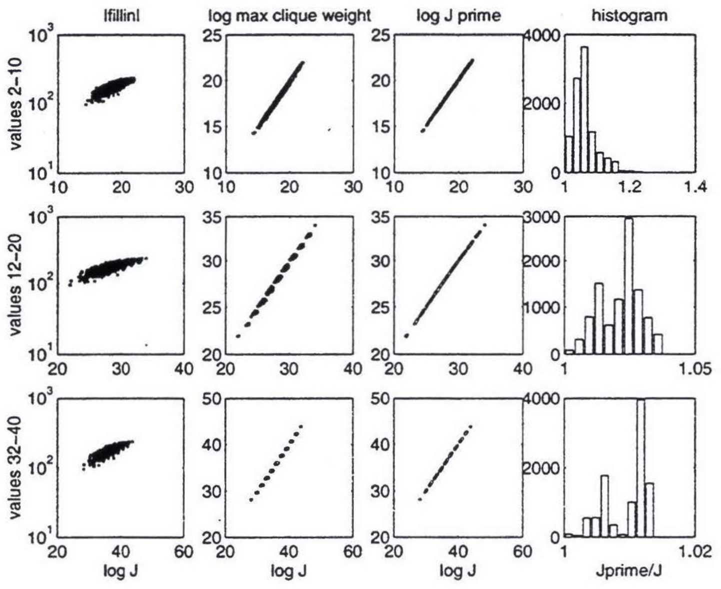

Figure 1: Size of the fill-in, maximum clique and raw weight J' versus J; histogram of J'/J for 10,000 triangulations of a a 30 node, 87 edges DAG. The r_v ranges are 2-10, 12-20, 32-40; the respective upper bounds for J'/J given by (5) are 2., 1.091 and 1.032. "Line" thinner means better agreement with J.

To constrain the optimum to be a permutation, we add the penalty term $\lambda \sum_{ij} \theta_{ij} \log \theta_{ij}$ where $\lambda > 0$ is a parameter that is progressively increased following a deterministic annealing schedule.

4 Simulation results

Simulations were performed to explore the usefulness of J' as a cost function and to assess the performance of our algorithm.

For the first goal, we generated random directed acyclic graphs (DAGs) of different sizes and densities[4] on which we computed the values of J and J' for 50,000 random triangulations. For each of them, table 1 syntesizes the results in terms of J'/J. It can be seen that the typical values are much lower than the theoretical bound $1 + 1/(r_{min} - 1)$ and increase only slowly with n.

In figure 1 we present the relationship between J^F, J^C, J' and J for a 30 node graph and various ranges for r_v. The plots confirm that J^F is a poor substitute for J. It can also be seen that J' has the best agreement with J in all cases with J^C as a close second. As predicted by 5 the agreement improves when r_v becomes larger. Regarding J^C, its increase in "coarseness" with increasing r_v is visible, whereas the expected improvement in the agreement with J for large r_v is not evident in the present simulations.

For our second goal, only preliminary results have been obtained so far. Some of the graphs used are shown in figure 2.

[4]We defined the density to be the ratio between $|E|$ and the maximum possible number of edges $n(n-1)/2$.

	density		
n	.05	.1	.2
10	$1.033 < 1.113 < 1.212$	$1.011 < 1.180 < 1.621$	$1.016 < 1.184 < 1.502$
20	$1.037 < 1.188 < 1.447$	$1.018 < 1.221 < 1.695$	$1.012 < 1.228 < 1.829$
30	$1.023 < 1.234 < 1.662$	$1.018 < 1.236 < 1.838$	$1.014 < 1.243 < 1.874$
40	$1.020 < 1.244 < 1.825$	$1.017 < 1.249 < 1.869$	$1.014 < 1.244 < 1.861$

Table 1: Median values of the minimum, median and maximum values of J'/J obtained for 50,000 triangulations $\star$ 100 random DAGs. r_v was random in the range 2-6, giving an upper bound of 2. The mean values for each graph were very close to the median values.

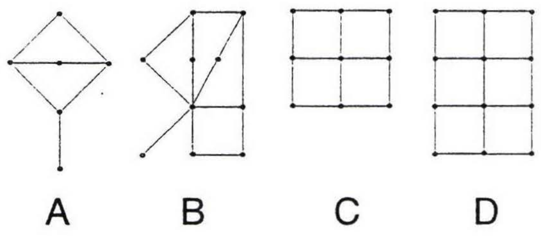

Figure 2: Test graphs for the triangulation algorithm. $r_v = 3 =$ constant.

Our algorithm produced an optimal solution for graphs A, B and C; for graph D, it produced a triangulation with a cost 10% higher than optimal. For this latter graph only the best instance of the minimum weight heuristic produced the optimal results. Lexicographic search and maximum cardinality search [9] produced results which were 15% from the optimum even in their best instance.

5 Discussion

Computing the objective J' and its derivatives w.r.t. θ requires a total of $\mathcal{O}(n^4)$ operations and $\mathcal{O}(\max[n^2, (\frac{n^2}{2} - m)n])$ storage, where m is the number of edges of G. Thus, we need a storage varying between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ depending upon the density of the graph. The most computationally intensive step is computing μ from θ by formula (3). We are investigating the possibility of parametrizing the problem directly in terms of μ which would reduce the computation cost to $\mathcal{O}(n^3 \log n)$. Replacing the actual expression of e with a smooth function that can be computed in reasonable time is also desirable.

In a practical implementation, a stage of pre-processing should precede the application of the algorithm. Pre-processing is aimed at pruning the graph of certain nodes and edges that are known not to affect the optimal triangulation. Examples thereof are bridge[5] removal and simplicial node [3] elimination. A *simplicial* node together with its neighbors forms a clique. Simplicial nodes can be eliminated recursively in any order before any other optimization takes place.

In the present paper we have proposed a method of transforming the combinatorial problem of finding an optimal permutation into a continuous optimization problem. Since in this process the initial cost function has been continuously extended over the

[5]A *bridge* is an edge that, when removed, increases the number of connected components of the graph.

domain Θ, minimizing it with the constraint that the solution lies in an extreme point should theoretically produce an optimal permutation with respect to J'. The cost function proposed here has been shown to agree well with the true objective J and is easy to compute in the conjunction with any elimination procedure. Therefore it may prove useful in contexts much broader than the one of the present method.

Acknowledgements

The authors are grateful to Tommi Jaakkola for many stimulating discussions.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[2] M.L. Balinski and R. Russakoff. On the assignment polytope. *SIAM Rev.*, 1974.

[3] Ann Becker and Dan Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *UAI 96 Proceedings (to appear)*, 1996.

[4] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.

[5] U. Kjærulff. Triangulation of graphs–algorithms giving small total state space. Technical Report Technical Report R 90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.

[6] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 1991.

[7] Marina Meila and Michael Jordan. Optimal triangulation with continuous cost functions. *Submitted to NIPS '96*, 1996.

[8] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 1970.

[9] D. J. Rose, R. E. Tarjan, and E.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Comput.*, 1976.

[10] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Math.*, 1981.