MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1605 March, 1997
C.B.C.L. Memo No. 146

Triangulation by Continuous Embedding

Marina Meilă **Michael I. Jordan**

Abstract

When triangulating a belief network we aim to obtain a junction tree of minimum state space. According to [8], searching for the optimal triangulation can be cast as a search over all the permutations of the network's variables. Our approach is to embed the discrete set of permutations in a convex continuous domain D. By suitably extending the cost function over D and solving the continous nonlinear optimization task we hope to obtain a good triangulation with respect to the aformentioned cost. In this paper we introduce an upper bound to the total junction tree weight as the cost function. The appropriatedness of this choice is discussed and explored by simulations. Then we present two ways of embedding the new objective function into continuous domains and show that they perform well compared to the best known heuristic.

This report describes research done at the Dept. of Electrical Engineering and Computer Science, the Dept. of Brain and Cognitive Sciences, the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Dept. of Defense and by the Office of Naval Research. Michael I. Jordan is a NSF Presidential Young Investigator. The authors can be reached at M.I.T., Center for Biological and Computational Learning, 45 Carleton St., Cambridge MA 02142, USA. E-mail: mmp@ai.mit.edu, jordan@psyche.mit.edu

1 Introduction. What is triangulation ?

Belief networks are graphical representations of probability distributions over a set of variables. For an introductory, yet rigorous treatment of graphical probability models, the reader is refered to [5]. In what follows it will be always assumed that the variables take values in a finite set and that they correspond to the vertices of a graph. The graph's arcs represent the dependencies among variables. There are two kinds of representations that have gained wide use: one is the directed acyclic graph model, also called a *Bayes net*, which represents the joint distribution as a product of the probabilities of each vertex conditioned on the values of its parents; the other is the undirected graph model, also called a *Markov field*, where the joint distribution is factorized over the *cliques*[1] of an undirected graph. This factorization is called a *junction tree* and optimizing it is the subject of the present paper. The power of both models lies in their ability to display and exploit existent marginal and conditional independencies among subsets of variables. Emphasizing independencies is useful from both a qualitative point of view (it reveals something about the domain under study) and a quantitative one (it makes computations tractable). The two models differ in the kinds of independencies they are able to represent and often times in their naturalness in particular tasks. Directed graphs are more convenient for learning a model from data; on the other hand, the clique structure of undirected graphs organizes the information in a way that makes it immediately available to inference algorithms. Therefore it is a standard procedure to construct the model of a domain as a Bayes net and then to convert it to a Markov field for the purpose of querying it.

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 [7] is constructed from the maximal cliques 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 state space. The *weight of the junction tree* is the sum of the weights of its component cliques. All further exact inference in the net takes place in the junction tree representation. 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 a suitably chosen continuous space. Then extend the cost to a smooth function over the continuous domain and thus transform the discrete optimization problem into a continuous nonlinear optimization task. This allows one 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 methods 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 junction 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 $v \in V$ in the triangulated graph and by $C_v = \{v\} \cup \{u \in \mathrm{n}(v) \mid \#u > \#v\}$[2], (e.g. the clique that formes by the elimination of v). Then, a result in [4] allows us to express the total weight of the junction tree obtained with elimination ordering $\#$ as

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

where $\mathrm{ismax}(C_v)$ 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

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

[2] Both $\mathrm{n}(v)$ and C_v depend on $\#$ but we chose not to emphasize this in the notation for the sake of readability.

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 [6]), 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* (MW) heuristic [6], a greedy minimization method for J^* (thus taking r_v into account) and later a simulated annealing approach [7] 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.

$$J^C \;=\; \max_C \prod_{u \in C} r_u. \tag{4}$$

J^C is *coarser* than J^* in the sense that triangulations with different values of J^* can have the same cliqueweight. 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 it instead 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 $r = \min r_v$ the following inequalities hold:

$$\frac{1}{n-1} J^*_{(\#)} \;\le\; J^C_{(\#)} \;\le J^*_{(\#)} \tag{5}$$

$$J^*_{(\#)} \;\le\; J_{(\#)} \;\le\; J^*_{(\#)} \frac{r}{r-1}\left(1 - \frac{1}{r^n}\right) \tag{6}$$

J^C is always lower bounding J^*, with equality in the case of a fully connected graph. As for J, by (6) it is less than a fraction $1/(r-1)$ away from J^*. The upper 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 triangulations, where our search will be concentrated. 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. Think for example of the tree represented in figure 1. A tree is an already triangulated graph, so its best triangulation contains no fill-in edges and comprises $n-1$ maximal cliques of size 2. However, it's worst triangulation, the fully connected graph, will have only 1 maximal clique of size n. But the former situation - many small maximal cliques - means that there will be only a few non-maximal cliques of small size to contribute to the difference $J - J^*$, and therefore that the agreement with J^* is usually closer than (6) implies. The simulations in section 4 will further support our choice.

Note also that the reverse argument holds for J^C: the maximum clique size is an inaccurate approximation of $J^*_{(\#)}$ for low cost triangulations but it is close to the true cost function for the worse ones, that produce a small number of large cliques.

3 The continuous optimization problem

This section shows two ways of defining J over continuous domains. Both rely on a formulation of J that eliminates explicit reference to the cliques C_v and that will be deduced now.

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

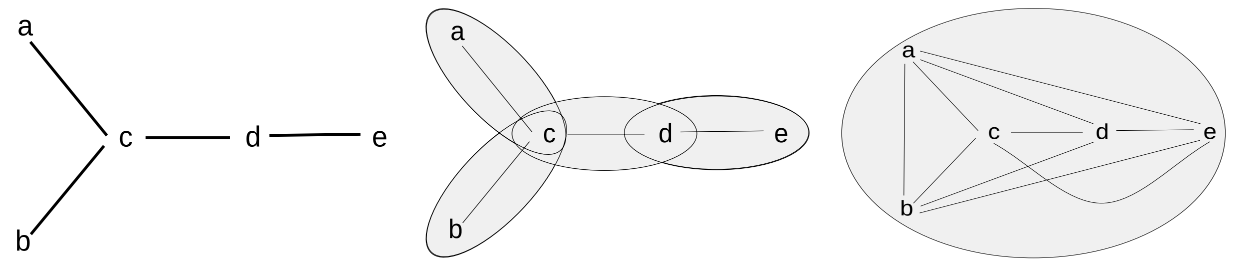
$$\mu_{uv} \;=\; \begin{cases} 1 & \text{if } \#u \le \#v \\ 0 & \text{otherwise} \end{cases}$$

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

In other words, μ represent precedence relationships and e represent the edges between the n vertices after the elimination. Therefore, they will be called *precedence* variables and *edge* variables respectively. With these variables, J can be expressed as

$$J_{(\#)} \;=\; \sum_{v \in V} \prod_{u \in V} r_u^{\mu_{vu} e_{vu}} \tag{7}$$

The product $\mu_{vu} e_{vu}$ acts as an indicator variable being 1 iff "$u \in C_v$" is true. For any given permutation, finding the μ variables is straightforward. It remains to show how to compute the e variables, or, in other words, how to find in advance, based on the precedence variables only, if a certain edge not in E will appear after eliminating the vertices in a given order. This is possible, thanks to a result in [9]. It states that an edge (u,v) is contained in $F_\#$ iff there is a path in G between u and v containing only nodes w for which $\#w < \min(\#u, \#v)$. Formally, $e_{uv} = e_{vu} = 1$ iff there exists a path $P = (u, w_1, w_2, \ldots v)$



a graph
$r_v = r$
$n = 5$

a

a good triangulation
$J^C = r^2$
$J^* = (n-1)r^2$
$J = (n-1)r^2 + r$

b

a bad triangulation
$J^C = r^n$
$J^* = r^n$
$J = r^n + r^{n-1} + \ldots + r$
$< r^n \frac{1}{r-1}$

c

Figure 1: Example illustrating inequalities (5) and (6). (b) and (c) are two triangulations of the graph represented in (a). For the triangulation in (b), J^* is at its optimum, J is only slightly larger and J^C is much smaller (by a factor of $1/(n-1)$) than J^*; for the fully connected triangulation represented by (c), J^* is at its maximum and J^C, also at its maximum, is equal to it, whereas J is at the maximum possible distance (but less than a factor of 2 away) from J^*.

such that
$$\prod_{w_i \in P} \mu_{w_i u} \mu_{w_i v} = 1$$

So far, we have succeeded to define the cost J associated with any permutation in terms of the variables μ and e. In the following, the set of permutations will be embedded in a continuous domain. As a consequence, μ and e will take values in the interval $[0, 1]$ but the form of J in (7) will stay the same.

The first method, called μ-continuous embedding (μ-CE) assumes that the variables $\mu_{uv} \in [0, 1]$ represent independent probabilities that $\#u < \#v$. To make an assignment of the μ variables represent a permutation, we need to ensure that it is antisymmetric and transitive. Antisymmetry means that $\#v < \#w$ and $\#w < \#v$ cannot be true in the same time and it is guaranteed by definition. Transitivity means that if $\#u < \#v$ and $\#v < \#w$, then $\#u < \#w$, or, that for any triple $(\mu_{uv}, \mu_{vw}, \mu_{wu})$ the assignments $(0, 0, 0)$ and $(1, 1, 1)$ are forbidden. We will penalize the possible violations of transitivity using the probabilistic interpretation of μ. According to it, we introduce a penalty term that upper bounds the probability of a nontransitive triplet:

$$\begin{aligned} R(\mu) &= \sum_{u<v<w} P[(u, v, w) \text{ nontransitive}] \qquad (8) \\ &= \sum_{u<v<w} [\mu_{uv}\mu_{vw}\mu_{wu} + (1-\mu_{uv})(1-\mu_{vw})(1-\mu_{wu})] \\ &\geq P[\text{assignment non transitive}] \end{aligned}$$

Recovering the permutation from a μ assignment is easily done noting that

$$0 \leq \#v - 1 = \sum_{u \neq v} \mu_{uv} \leq n - 1 \qquad (9)$$

The formula can be used for non-integer μ values as well.

In the second approach, called θ-continuous embedding (θ-CE), the permutations are directly embedded into the set of doubly stochastic matrices. A *doubly stochastic matrix* θ is a matrix for which the elements in a row or column sum to one.

$$\sum_i \theta_{ij} = \sum_j \theta_{ij} = 1 \quad \theta_{ij} \geq 0 \quad \text{for } i, j = 1, ..n. \qquad (10)$$

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*. $\theta_{ij} = 1$ and $\#i = j$ 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. To constrain the optimum to be a an extreme point, we add the penalty term

$$R(\theta) = \sum_{ij} \theta_{ij}(1 - \theta_{ij}) \qquad (11)$$

The precedence variables are defined over Θ as

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

In both embeddings, the edge variables are computed from μ as follows

$$e_{uv} = e_{vu} = \begin{cases} 1, & \text{for } (u, v) \in E \text{ or } u = v \\ \max\limits_{P \in \{paths\ u \to v\}} \prod_{w \in P} \mu_{wu}\mu_{wv}, & \text{otherwise} \end{cases} \qquad (12)$$

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 $(w_1, w_2) \in E$ defined as $(-\log \mu_{w_1 u}\mu_{w_2 v})$.

	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 $\times$ 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.

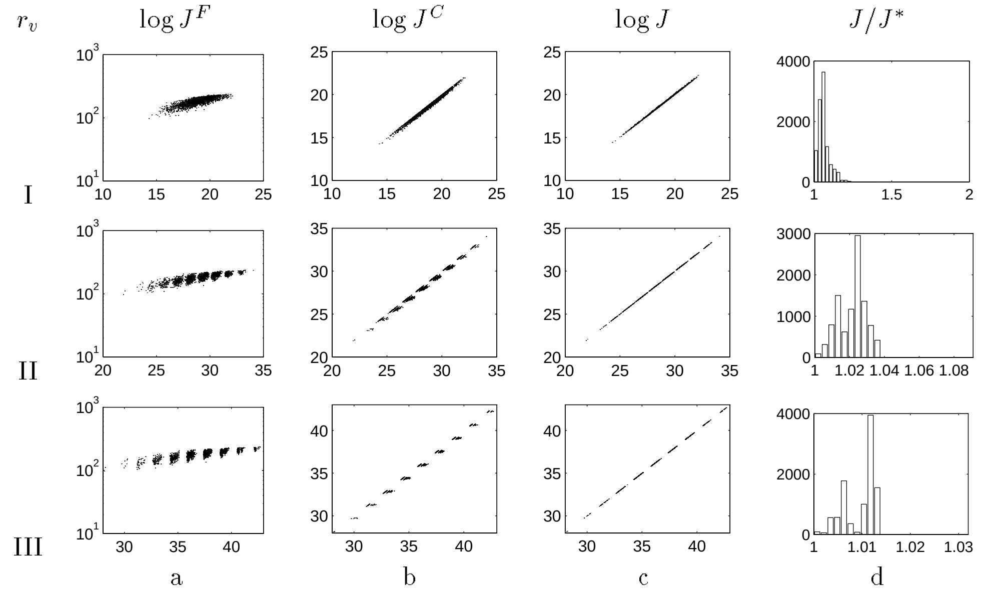


Figure 2: Size of the fill-in J^F (a), maximum clique J^C (b) and raw weight J (c) versus J^*; histogram of J/J^* (d) for 10,000 triangulations of a 30 node, 87 edges DAG. The r_v values are uniform in the ranges 2-10 (I), 12-20 (II), 32-40 (III). For the histograms in (d) the right limit of the plot is placed at the theoretical upper bound $r_{\min}/(r_{\min}-1)$. For (a), (b), (c), "line" thinner means better agreement with J^*.

θ-CE is an interior point method whereas in μ-CE the current point, although inside $[0,1]^{n(n-1)/2}$ isn't necessarily in the convex hull of the hypercube's corners that represent permutations. The number of operation required for one evaluation of J and its gradient is as follows: $\mathcal{O}(n^4)$ operations to compute μ from θ, $\mathcal{O}(n^3 \log n)$ to compute e, $\mathcal{O}(n^3)$ for $\frac{\partial J}{\partial e}$ and $\mathcal{O}(n^2)$ for $\frac{\partial J}{\partial \mu}$ and $\frac{\partial J}{\partial \theta}$ afterwards. Since computing μ is the most computationally intensive step, μ-CE is a clear win in terms of computation cost. In addition, by operating directly in the μ domain, one level of approximation is eliminated, which makes us expect it to perform better than θ-CE. This expectation will be confirmed by the results in the next section.

4 Experimental results

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

For the first goal, we generated random directed acyclic graphs (DAGs) of different sizes and densities[3] 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 $1+1/(r_{min}-1)$. The large values tend to spread towards the upper bound with the increase of the number of vertices n, but the median and lowest values increase much slower if at all, suggesting that the agreement between J and J^* is better than theoretically predicted over a wide range of graph sizes, densities and structures.

In figure 2 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 second. As predicted by (6) the agreement improves when r_v becomes larger. Regarding J^C, its increase in "coarseness" with increasing r_v is reflected by the "step-like" appearance aspect

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

graph	**h9**	**h12**	**d10**	**m20**	**a20**	**d20**
$n = \|V\|$	9	12	10	20	20	20
density	.33	.25	.6	.25	.45	.6
$r_{\min}/r_{\max}/r_{\text{avg}}$	2/2/2/	3/3/3	6/15/10	2/8/5	6/15/10	6/15/10
$\log_{10} J^*_{MW}$	2.43	2.71	7.44	5.47	12.75	13.94

Table 2: Characteristics of the graphs used in the experiments.

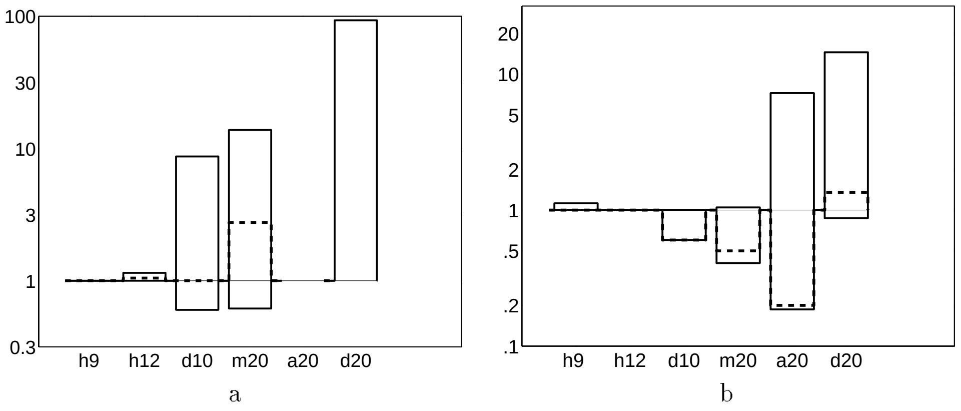


Figure 3: Minimum, maximum (solid line) and median (dashed line) values of $\frac{J^*}{J^*_{MW}}$ obtained by θ-CE (a) and μ-CE (b).

of J^C, whereas the expected improvement in the agreement with J^* for large r_v is not evident in the present simulations.

For the second goal we compared the results of our algorithms with the results of the minimum weight heuristic (MW), the heuristic that scored best in empirical tests [6]. The lowest junction tree weight obtained in 200 runs of MW was retained and denoted by J^*_{MW}. Tests were run on 6 graphs of different sizes and densities shown in table 2.

We ran 11 or more trials of each of our two algorithms on each graph. To enforce the variables to converge to a permutation, we minimized the objective $J + \lambda R$, where $\lambda > 0$ is a parameter that was progressively increased following a deterministic annealing schedule and R is one of the aforementioned penalty terms. The algorithms were run for 50-150 optimization cycles, usually enough to reach convergence. However, for the μ-embedding on graph **d20**, there were several cases where many μ values did not converge to 0 or 1. In those cases we picked the most plausible permutation to be the answer.

The results are shown in figure 3 in terms of the ratio of the true cost obtained by the continuous embedding algorithm (denoted by J^*) and J^*_{MW}. For the first two graphs, **h9** and **h12**, J^*_{MW} is the optimal cost; the embedding algorithms reach it most trials. On the remaining graphs, μ-CE clearly outperforms θ-CE, which also performs poorer than MW on average. On **d10**, **a20** and **m20** it also outperforms the MW heuristic, attaining junction tree weights that are 1.6 to 5 times lower on average than those obtained by MW. On **d20**, a denser graph, the results are similar for MW and μ-CE in half of the cases and worse for μ-CE otherwise. The plots also show that the variability of the results is much larger for CE than for MW. This behaviour is not surprising, given that the search space for CE, although continuous, comprises a large number of local minima. This induces dependence on the initial point and, as a consequence, nondeterministic behaviour of the algorithm. Moreover, while the number of choices that MW has is much lower than the upper limit of $n!$, the "choices" that CE algorithms consider, although soft, span the space of all possible permutations.

5 Discussion

The idea of continuous embedding is not new in the field of applied mathematics. The large body of literature dealing with smooth (sygmoidal) functions instead of hard nonlinearities (step functions) is only one example. The present paper shows a nontrivial way of applying a similar treatment to a new problem in a new field. The results obtained by μ-embedding are on average better than the standard MW heuristic. Although not directly comparable, the best results reported on triangulation [7, 3] are only by little better than ours. Therefore the significance of our results goes beyond the scope of the present problem. They are obtained on a hard problem, whose cost function has no feature to ease its minimization (J is neither linear, nor quadratic, nor is it additive w.r.t. the vertices or the edges) and thus they demonstrate the potential of continuous embedding as a general tool.

The cost function that we have introduced, J, has the twofold advantage of being more accurate than all the other alternative cost functions used in the literature and of being directly amenable to continuous approximations. Since minimizing J may not be NP-hard, this opens a way for investigating new triangulation methods.

Acknowledgements

The authors are grateful to Tommi Jaakkola for many stimulating discussions and to David Maze for participating in the implementation of the algorithms.

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*, 1996.

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

[5] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistical Quarterly*, 1990.

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

[7] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 1992.

[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.