Approximate Inference in Graphical Models using Tensor Decompositions

Marcel van Gerven
Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
marcelge@cs.ru.nl

Abstract

We demonstrate that tensor decompositions can be used to transform graphical models into structurally simpler graphical models that approximate the same joint probability distribution. In this way, standard inference algorithms such as the junction tree algorithm, can be used in order to use the transformed graphical model for approximate inference. The usefulness of the technique is demonstrated by means of its application to thirty randomly generated small-world Markov networks.

Key words: graphical models; approximate inference, tensor decompositions

1 Introduction

Graphical models such as Bayesian networks and Markov networks allow for probabilistic inference using a graph representation that factorizes a joint probability distribution. In case the graph structure becomes too dense, exact inference becomes intractable, and in those cases we may resort to approximate inference algorithms. These algorithms can be distinguished into *stochastic* and *deterministic* methods. Examples of stochastic approximate inference methods are *importance sampling* and *Gibbs sampling* [7]. Examples of deterministic approximate inference methods are *loopy belief propagation* [16], which is the application of belief propagation [17] to acyclic directed graphs, and *variational methods* [10], which transform a graphical model into a less complex model in order to compute bounds on probabilities of interest.

In this paper, we follow an approach that is reminiscent of variational methods in the sense that we transform a graphical model into a less complex model. However, in our case, this transformation is not done analytically by means of computing bounds on quantities of interest, but proceeds by transforming graphs into less complex graphs using the machinery of tensor decompositions [21, 3, 8]. Using these tensor decompositions, approximate inference can be realized by means of marginalization over additional hidden variables. Since the approximation is accomplished by means of a structural transformation it allows the subsequent use of exact inference algorithms such as the junction tree algorithm and variants thereof [12, 19, 13].

This approach to approximate inference in arbitrary graphical models expands on earlier work [18], which has demonstrated the application of tensor decompositions for graphical models that employ the concept of independence of causal influence [6], such as the well-known noisy-or model [17].

This paper proceeds as follows. We begin by introducing tensors, tensor operations, and tensor decompositions in Section 2. Subsequently, we move on to discuss probabilistic inference in graphical models in Section 3, where we focus on the well-known junction tree algorithm. The main contribution of this paper is formed by Section 4, where we show how tensor decompositions allow for approximate inference in graphical models. This is demonstrated by means of the decomposition of thirty randomly generated Markov networks whose underlying graph structure resembles that of real-world graphs in Section 5. We end this paper in Section 6 with a discussion of the proposed technique.

2 Tensors: Operations and Decompositions

In the following, we use calligraphic characters $\mathcal{A}$ to denote tensors, uppercase boldface characters $\mathbf{A}$ to denote matrices, and lowercase boldface characters $\mathbf{a}$ to denote vectors. Lowercase standard characters are used to indicate elements of tensors, matrices, or vectors.

2.1 Tensors

A tensor is a concept taken from multi-linear algebra which generalizes the concepts of vectors and matrices, and is defined as follows.

Definition 2.1. *Let $I_1, \ldots, I_N \in \mathbb{N}$ denote index upper bounds. A* tensor *$\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ of* order *N is an N-way array where elements $a_{i_1 \cdots i_n}$ are indexed by $i_j \in \{1, \ldots, I_j\}$ for $1 \leq j \leq N$.*

Hence, a tensor of order one denotes a vector $\mathbf{a} \in \mathbb{R}^{I_1}$, and a tensor of order two denotes a matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$. In the following we introducing the theoretical background that is required to understand the concept of a tensor decomposition.

2.2 Tensor operations

A tensor can be expressed in terms of a matrix using the concept of a matrix unfolding.

Definition 2.2. *The* matrix unfolding *$\mathbf{A}_{(j)} \in \mathbb{R}^{I_j \times (I_{j+1} I_{j+2} \cdots I_N I_1 I_2 \cdots I_{j-1})}$ of an Nth order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the matrix that has element $a_{i_1 \cdots i_N}$ at row number i_j and column number*

$$1 + \sum_{\substack{1 \leq k \leq N \\ k \neq j}} (i_k - 1) \prod_{\substack{k+1 \leq m \leq N \\ m \neq j}} I_m \, .$$

Example 2.1. The matrix unfolding $\mathbf{A}_{(2)}$ of a third-order tensor

$$\mathcal{A} = \begin{pmatrix} (a,b)^T & (c,d)^T \\ (e,f)^T & (g,h)^T \end{pmatrix} \text{ is given by } \mathbf{A}_{(2)} = \begin{pmatrix} a & b & e & f \\ c & d & g & h \end{pmatrix} .$$

The nth *mode* of a tensor refers to the nth dimension of a tensor. A tensor may be multiplied by a matrix by means of the *n-mode product.*

Definition 2.3. *The* n-mode product $\mathcal{A} \times_n \mathbf{B}$ *of a tensor* $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *and a matrix* $\mathbf{B} \in \mathbb{R}^{J_N \times I_N}$, *is a tensor* $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}$ *with elements:*

$$c_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_N} b_{j_n i_n} \,.$$

Example 2.2. Let $\mathcal{A}$ be a third-order tensor as in example 2.1 and let $\mathbf{B}$ denote a square matrix with $b_{11} = u$, $b_{12} = v$, $b_{21} = w$, $b_{22} = x$. The 2-mode product $\mathcal{A} \times_2 \mathbf{B}$ is then given by

$$\begin{pmatrix} (a(u+v), b(u+v))^T & (c(w+x), d(w+x))^T \\ (e(u+v), f(u+v))^T & (g(w+x), h(w+x))^T \end{pmatrix} .$$

We also define for tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, the *inner product*

$$\langle \mathcal{A}, \mathcal{B} \rangle \equiv \sum_{i_1, \ldots, i_N} a_{i_1 \cdots i_N} b_{i_1 \cdots i_N}$$

and *Frobenius norm*

$$|| \mathcal{A} || \equiv \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}.$$

The *outer product* of two tensors is defined as follows.

Definition 2.4. *The* outer product $\mathcal{A} \circ \mathcal{B}$ *of two tensors* $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_n}$ *and* $\mathcal{B} \in \mathbb{R}^{J_1 \times \cdots \times J_n}$ *is defined as the tensor* $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times J_1 \times \cdots \times J_n}$ *such that*

$$c_{i_1 \cdots i_n j_1 \cdots j_n} = a_{i_1 \cdots i_n} \cdot b_{j_1 \cdots j_n}$$

for all elements of $\mathcal{C}$.

Using the outer product, the *rank* of a tensor is defined as follows [9].

Definition 2.5. *A tensor of order* N *has* rank *one if it can be written as an outer product* $\mathbf{a}^{(1)} \circ \cdots \circ \mathbf{a}^{(N)}$ *of vectors. The rank of a tensor* $\mathcal{A}$ *is defined as the minimal number of tensors* $\mathcal{A}_1, \ldots, \mathcal{A}_K$ *of rank one such that*

$$\mathcal{A} = \sum_{k=1}^{K} \mathcal{A}_k \,. \tag{1}$$

Example 2.3. The third-order tensor

$$\mathcal{A} = \begin{pmatrix} (\ \ \ 6, -3)^T & (\ \ \ 8, -4)^T \\ (-12, \ \ 6)^T & (-16, \ \ 8)^T \end{pmatrix}$$

has rank one since it can be written as the outer product of $(1, -2)^T$, $(3, 4)^T$, and $(2, -1)^T$.

2.3 Tensor decompositions

Equation (1) shows that a tensor may be written in terms of a sum of K rank one tensors, which is essentially a *rank-K decomposition* of $\mathcal{A}$. This rank-K decomposition is a special case of the *Tucker decomposition* [21]:

$$T_{\mathbf{J}}(\mathcal{A}) = \mathcal{C} \times_1 \mathbf{B}^{(1)} \times_2 \cdots \times_N \mathbf{B}^{(N)} \tag{2}$$

with $\mathbf{J} = (J_1, \ldots, J_N)$, a *core tensor* $\mathcal{C} = (c_{j_1 \cdots j_N})$ and matrices $\mathbf{B}^{(n)} \in \mathbb{R}^{I_n \times J_n}$. Elements of the original tensor $\mathcal{A}$ are then computed as follows:

$$a_{i_1 \cdots i_N} = \left(\sum_{j_1, \ldots, j_N} c_{j_1 \cdots j_N} \cdot b^{(1)}_{i_1 j_1} \cdots b^{(N)}_{i_N j_N} \right) + r_{i_1 \cdots i_N} \,, \tag{3}$$

where $(r_{i_1 \cdots i_N})$ denotes a residual tensor $\mathcal{R}$. When we assume that the core tensor $\mathcal{C}$ is a super-diagonal tensor with $c_{j_1 \cdots j_N}$ equal to one if $j_1 = j_2 = \cdots = j_N$ and zero otherwise, then we obtain:

$$a_{i_1 \cdots i_N} = \left(\sum_{k=1}^{K} \lambda_k \cdot b^{(1)}_{i_1 k} \cdots b^{(N)}_{i_N k} \right) + r_{i_1 \cdots i_N} \tag{4}$$

which reduces to the rank-K decomposition for vanishing $r_{i_1 \cdots i_N}$. Equation (4) is also known as the *canonical decomposition* [3], or *parallel factors decomposition* [8]. We will refer to it as a rank-K approximation of a tensor $\mathcal{A}$; also written as $R_K(\mathcal{A})$.

One way to find a rank-1 approximation is by means of the *higher-order power method* (HOPM) [5], as shown in Algorithm 1. This alternating least squares algorithm finds a tensor $\hat{\mathcal{A}} = \lambda \cdot \mathbf{b}^{(1)} \circ \cdots \circ \mathbf{b}^{(N)}$, with scalar λ and unit-norm vectors $\mathbf{b}^{(n)}$, $1 \leq n \leq N$, minimizing the least-squares cost function $C(\mathcal{A}, \hat{\mathcal{A}}) \equiv \| \mathcal{A} - \hat{\mathcal{A}} \|^2$. In order to initialize matrices and vectors in Algorithm 1, various schemes can be used. One approach is to repeat the algorithm for several random initializations and to choose that decomposition which maximizes the fit between the original tensor and the approximation. Another approach that has been shown to give good results, and which is used in this paper, is to choose the first dominant left singular vector of the matrix unfolding $\mathbf{A}_{(j)}$, as an initial estimate of $\mathbf{b}^{(j)}$ [5, 4].

A greedy approach to finding a rank-K approximation is to apply the higher-order power method to the residuals that remain after obtaining a rank-1 approximation. This technique has been employed successfully in Ref. [23] in order to achieve high compression rates for image sequences.

```
input: $\mathcal{A}$
initialize $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(N)}$
repeat
  for $n = 1$ to $N$ do
    $\tilde{\mathbf{b}}^{(n)} = \mathcal{A} \times_1 \mathbf{b}^{(1)^T} \times_2 \cdots \times_{n-1} \mathbf{b}^{(n-1)^T} \times_{n+1} \mathbf{b}^{(n+1)^T} \times_{n+2} \cdots \times_N \mathbf{b}^{(N)^T}$
    $\lambda_n = \| \tilde{\mathbf{b}}^{(n)} \|$
    $\mathbf{b}^{(n)} = \tilde{\mathbf{b}}^{(n)} / \lambda_n$
  end for
until convergence
return $\hat{\mathcal{A}} = \lambda_N \cdot \mathbf{b}^{(1)} \circ \cdots \circ \mathbf{b}^{(N)}$
```

Algorithm 1: The higher-order power method for finding a rank-1 approximation of a tensor.

By defining $\mathcal{A}^1 \equiv \mathcal{A}$ and $\mathcal{A}^k \equiv \mathcal{A}^{k-1} - \text{HOPM}(\mathcal{A}^{k-1})$ the following rank-K approximation of a tensor $\mathcal{A}$ is obtained:

$$R_K(\mathcal{A}) \equiv \sum_{k=1}^{K} \text{HOPM}(\mathcal{A}^k)\,. \tag{5}$$

The rank-K approximation of Eq. (4) as computed from Eq. (5) has an important interpretation in the context of approximate inference in graphical models, as we will show in Section 4. First, however, we will turn towards probabilistic inference in graphical models in general.

3 Probabilistic Inference in Graphical Models

Graphical models represent independence between random variables by means of a graph. If the graph is directed and acyclic then the graphical model is known as a Bayesian network, and if it is undirected then it is known as a Markov network. Since a Bayesian network can be transformed into a Markov network by means of a moralization operation that connects parents and drops arc orientation, we will focus our attention on Markov networks.

Definition 3.1. *A* Markov network $\mathcal{M} = (G, \Psi)$ *is a pair, where G is an* undirected graph *with nodes corresponding to a set of random variables $\mathbf{X}$ and $\Psi = \{\psi_i(\mathbf{c}_i) \colon \mathbf{C}_i \in \mathcal{C}\}$ is a set of non-negative functions, known as* potentials*, defined for the maximal cliques (maximally complete subgraphs) $\mathcal{C}$ of G.*

By representing a joint probability distribution (JPD) in terms of a product of local factors, a Markov network with cliques $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_m\}$ allows the following factorization:

$$P(\mathbf{x}) = \frac{1}{Z}\prod_{i=1}^{m} \psi_i(\mathbf{c}_i) \tag{6}$$

where $Z = \sum_{\mathbf{x}} \prod_{i=1}^{m} \psi_i(\mathbf{c}_i)$ is the *partition function*, which acts as a normalizing constant.

Graphical models reduce the number of free parameters that are needed to specify a JPD, thereby allowing efficient probabilistic inference, such as the computation of conditional and marginal probabilities for random variables $\mathbf{U} \subseteq \mathbf{X}$ given evidence $\mathbf{E} \subseteq \mathbf{X} \setminus \mathbf{U}$. Over the years, various exact and approximate inference methods have been developed, where exact methods typically require the graph structure underlying a graphical model to be sufficiently sparse.

The *junction tree algorithm* [12] is an exact inference algorithm that allows for the computation of conditional and marginal probabilities in arbitrary graphs. In this paper, we take the standard junction tree algorithm as our point of departure, and focus on discrete random variables. In order to apply the junction tree algorithm to a Markov network (G, Ψ), we need to ensure that G is *triangulated*. An undirected graph is said to be triangulated when all loops of length four or more have at least one edge between non-neighboring nodes. The cliques of a triangulated graph can be arranged to form a junction tree which satisfies the running intersection property: if a node appears in two cliques $\mathbf{C}$ and $\mathbf{C}'$, then it will also appear in all cliques that lie on the path between $\mathbf{C}$ and $\mathbf{C}'$. Figure 1 depicts (optional) moralization, triangulation, and transformation into a junction tree.

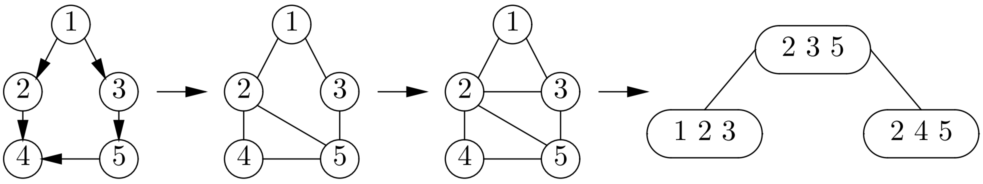

Figure 1: Transforming an acyclic digraph into a moralized and triangulated undirected graph and finally into a junction tree, satisfying the running intersection property.

Inference proceeds by means of evidence absorption and message passing in the junction tree, and produces posteriors for random variables $\mathbf{U}$. Since our interest is mainly in the structure of the Markov networks and junction trees, we will not elaborate on the exact form of the message passing protocol, and instead refer to [12].

The complexity of inference is exponential in the size of the largest clique after triangulation of the graph, and therefore, the aim is to obtain small clique sizes. The sizes of the cliques that are obtained after triangulation depend on the initial sizes of the cliques in the Markov network before marginalization and the chosen elimination ordering of the nodes during triangulation. Since finding an optimal elimination ordering is NP-complete [1], we use a greedy approach which, at each step, eliminates that node which will result in the addition of the least number of edges, where we break ties by choosing the node that induces the clique having the smallest weight [11]. The weight of a clique is defined as:

$$\mathrm{W}(\mathbf{C}) = \prod_{C \in \mathbf{C}} |\Omega_C|$$

where Ω_C is the state space of C. The weight of a Markov network is defined as the product of the weight of its cliques: $\mathrm{W}(\mathcal{M}) = \prod_{\mathbf{C} \in \mathcal{M}} \mathrm{W}(\mathbf{C})$. In the next section, we show how tensor decompositions can be used as the basis for approximate inference by reducing the weight of the used Markov networks and junction trees.

4 Approximate Inference using Tensors

In this section, we describe the use of tensor decompositions for approximate inference in graphical models. We first turn towards the interpretation of tensor decompositions in terms of graphical models.

4.1 Graphical model interpretation of tensor decompositions

Consider an arbitrary potential $\psi \colon \Omega_{\mathbf{C}} \to \mathbb{R}_0^+$ mapping configurations $\mathbf{c} \in \Omega_{\mathbf{C}}$ to positive real values. If random variables in $\mathbf{C} = \{X_1, \ldots, X_N\}$ are discrete, then we may interpret ψ as a tensor, such that $\psi_{x_1 \cdots x_N}$ denotes the positive real value associated with $(x_1, \ldots, x_N) \in \Omega_{\mathbf{C}}$. As described in Ref. [18], we may interpret a rank-K approximation in terms of a graphical model structure. According to Eq. (4), the rank-K approximation of ψ can be written as:

$$R_K(\psi)_{x_1 \cdots x_N} = \sum_{h=1}^{K} \lambda_h \cdot b^{(1)}_{x_1 h} \cdots b^{(N)}_{x_N k} \,. \tag{7}$$

By defining functions $\phi_j(x_j, h) \equiv b^{(j)}_{x_j h}$ for $1 \leq j < n$ and absorbing λ into the function $\phi_n(x_N, h) \equiv \lambda_h \cdot b^{(N)}_{x_N h}$, we obtain:

$$\psi(x_1, \ldots, x_N) \approx \sum_h \prod_{j=1}^{N} \phi_j(x_j, h)\,, \tag{8}$$

which can be interpreted as marginalization over a hidden variable H with states $h \in \Omega_H$, where we have dropped the requirement that potentials are non-negative. Since the hidden variable is marginalized out and the original potential is approximated by the marginalization, negative values cancel and we again obtain a non-negative potential. This interpretation is depicted in Fig. 2. Note that, in case the decomposition (8) uses just one component, it reduces to:

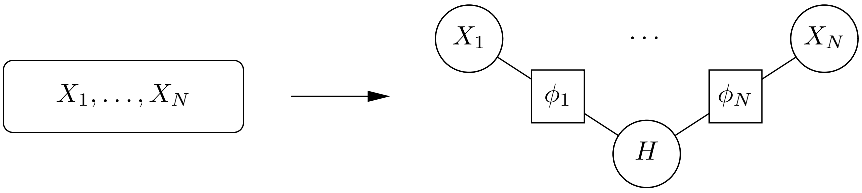

Figure 2: A clique $\{X_1, \ldots, X_N\}$ can be represented by a tensor rank-K approximation. This can be interpreted in terms of a graphical model, with (possibly negative) real-valued functions ϕ_j and a hidden variable H taking values in $\{1, \ldots, K\}$.

$$\psi(x_1, \ldots, x_N) \approx \prod_{j=1}^{N} \phi_j(x_j)\,, \tag{9}$$

which implies probabilistic independence between random variables X_i and X_j with $i, j \in \{1, \ldots, N\}, i \neq j$. In this case, random variables $X_1, \ldots, X_N$ are decoupled, which allows the representation of $\psi(x_1, \ldots, x_N)$ in terms of N potentials over single random variables.

4.2 Approximate inference with tensor decompositions

Savický and Vomlel focused in their work [18] on the exact decomposition of a restricted set of potentials that display functional dependence, which are tensors of the form $\psi(\mathbf{x}, y) = 1_{y=f(x)}$ for the indicator function 1_X and some function f. In this paper, in contrast, we focus on arbitrary potentials. The basic idea of approximate inference with tensor decompositions is to decompose the potentials in a Markov network into sets of smaller cliques using the rank-K approximation of Eq. (4).

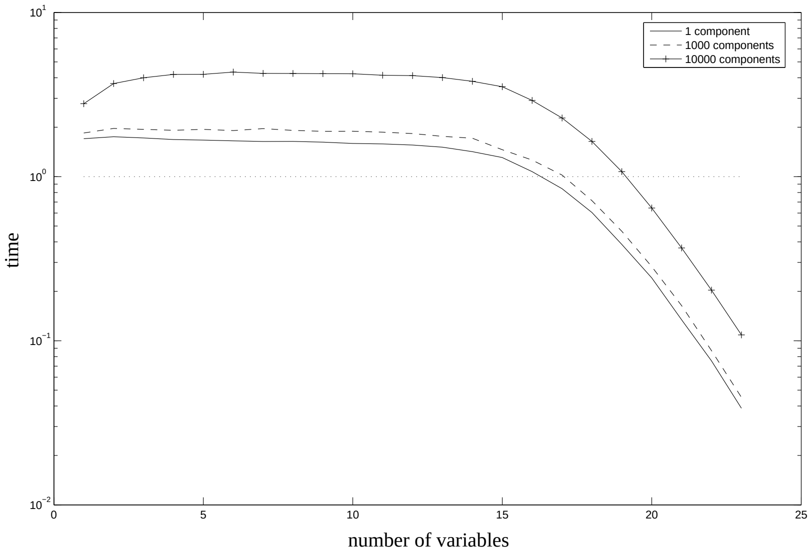

Figure 3: Inference time spent using a rank-K decomposition of a potential as compared with the original potential for binary variables and three different choices of K.

If we focus on a Markov network $\mathcal{M}$ consisting of one clique $\mathbf{C} = \{X_1, \ldots, X_N\}$ of binary random variables only, then the decomposition of $\mathbf{C}$ leads to a network weight of $2KN$ instead of 2^N, where K is the number of states of the hidden variable H. The smaller we choose K, the more efficient inference will become, whereas the higher we choose K, the better the approximation will become. The efficiency of this approximate inference method critically depends on how fast marginalization is for a decomposed potential as compared with marginalization for the non-decomposed potential. Figure 3 depicts the relative time that is spent on inference for different numbers of components K. It is shown that, with our implementation of the junction tree algorithm, inference in the decomposed potential becomes faster than inference in the original potential for a clique containing seventeen binary variables, given that one component is used in the decomposition. This is equivalent to a clique weight of about $1.3 \cdot 10^5$. For a Markov network consisting of one clique only, the approximate inference method is straightforward and outperforms standard inference whenever a critical weight (as shown in Fig. 3) is exceeded. In arbitrary Markov networks the situation is more complex since we need to take into account the triangulation step of the junction tree algorithm.

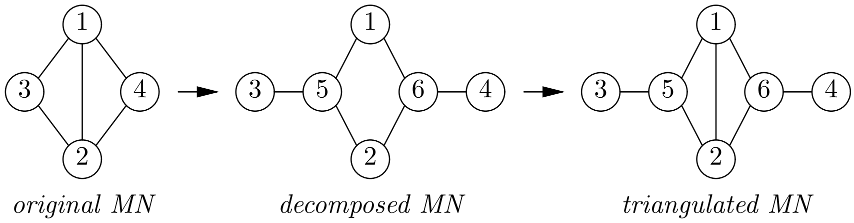

Figure 4: Example of a suboptimal application of decomposition.

Figure 4 depicts the problem associated with a naive decomposition of a Markov network, where we disregard for the moment the critical weight. The original Markov network consists of two cliques $\mathbf{C_1} = \{1,2,3\}$ and $\mathbf{C_2} = \{1,2,4\}$, and is already triangulated. After decomposition, the decomposed Markov network consists of six cliques of size two, but it is no longer triangulated! After triangulation, the triangulated decomposed Markov network has actually become more complex than the original Markov network, which invalidates the usefulness of the decomposition. In order for a Markov network decomposition to be useful, we need to make sure that the resulting decomposed Markov network after triangulation is significantly less complex than the original Markov network after triangulation, where we define complexity in terms of network weight. There are, however, some special cases, for which we can guarantee that a decomposition leads to more efficient inference.

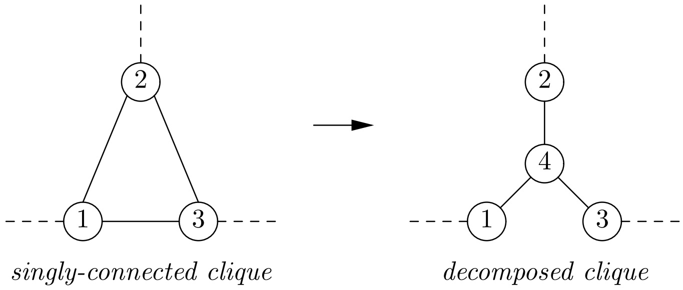

Figure 5: After decomposition, a singly-connected clique remains a triangulated component of the Markov network, since there is no loop containing nodes outside of a clique, that connects two distinct nodes within the clique.

One special case in which it is guaranteed that the decomposition is useful, is in case of a singly-connected clique, which is defined as a clique for which there is no loop containing nodes outside of a clique, that connects two distinct nodes within the clique (Fig. 5).

Proposition 4.1. *Decomposition of a singly-connected clique whose weight is above the critical weight always leads to more efficient inference.*

Proof. By construction, a singly-connected clique does not change after triangulation. Since its cliques after decomposition are triangulated as well, no edges will be added to the cliques of the decomposition, such that inference on the potential associated with the clique depends only on its critical weight. □

Another special case is due to the fact that a tensor decomposition which uses one component leads to a fully disconnected clique (Fig. 6).

Proposition 4.2. *Decomposition of a clique whose weight is above the critical weight using one component (state of the hidden variable H) only, never leads to less efficient inference.*

Proof. In the worst-case situation, the nodes that have been disconnected in the decomposed Markov network again become part of the same clique after triangulation, which does not induce extra overhead as compared with the original Markov network. □

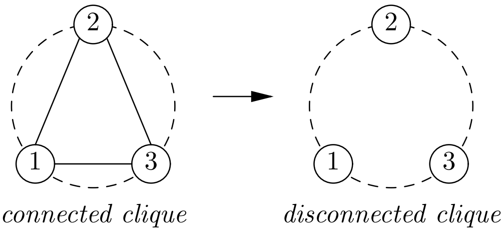

Figure 6: If the decomposition uses one component then we disconnect the clique. After triangulation, in the worst case, the original clique becomes connected again.

In this paper, we use a heuristic approach when determining the decomposition of a Markov network. We define $R^{\epsilon}(\psi) = \{\phi_1, \ldots, \phi_n\}$ as the rank-K decomposition of a potential $\psi(x_1, \ldots, x_n)$ into potentials $\{\phi_1(x_1, h)$, $\ldots$, $\phi_n(x_n, h)\}$, where the number of components K is such that the least-squares cost function $C(\psi, R_K(\psi)) < \epsilon$. Our approach for decomposing a Markov network is shown in Algorithm 2. It finds a decomposed Markov network that is approximately equal to the Markov network (depending on the choice of ϵ), and whose weight after triangulation is equal to or smaller than that of the original Markov network after triangulation due to the selective decomposition of potentials. Whether or not this leads to more efficient inference depends on the tradeoff between marginalization for a small

```
input: a Markov network $\mathcal{M} = (G, \Psi)$, an error criterion $\epsilon$
let $\mathcal{C} = \{\mathbf{C}_1, \ldots, \mathbf{C}_m\}$ denote the cliques of $G$.
let $\Psi_i$ denote the replacement of $\psi_i$ in $\Psi$ by $R^\epsilon(\psi_i)$
let $G_i$ denote the graph that is associated with $\Psi_i$
for $i = 1$ to $m$ do
  if $|\mathbf{C}_i| \geq 3$ and $\mathrm{W}(\mathrm{triangulate}(G_i, \Psi_i)) < \mathrm{W}(\mathrm{triangulate}(\mathcal{M}))$ then
    $\mathcal{M} = (G_i, \Psi_i)$
  end if
end for
return a decomposed Markov network $\mathcal{M}$
```

Algorithm 2: Algorithm for finding a decomposition of a Markov network.

number of large potentials and marginalization for a large number of small potentials, as was previously discussed in terms of critical weight.

5 Empirical Validation

The approximate inference method has been implemented in Matlab, where we have made use of both the Bayes Net Toolbox [15] and the Tensor Toolbox [2]. In order to validate our approximate inference method, we have tested it on randomly generated Markov networks. The graphs underlying the generated Markov networks are so-called *small-world networks* [24], which are highly clustered graphs that have a short characteristic path length. This type of graph shares similarities with real-world graphs, since many real-world graphs (such as social networks [14]) have a tendency to cluster, and, even though most nodes in the graph are not neighbors, most nodes can reach other nodes in a small number of steps. A small-world network is generated by reconnecting with probability p the edges in a regular ring lattice consisting of N nodes, with each node having K neighbors, as depicted in Fig. 7.

We have randomly generated thirty Markov networks, whose underlying graphs are small-world networks. We have chosen $N = 50$ as the number of nodes, $K = 8$ as the initial number of neighbors, and $p = 0.2$ as the reconnection probability. The potentials are given by random matrices and are defined over binary random variables. We have chosen three different values of the error criterion ($\epsilon = 1$, $\epsilon = 0.1$, and $\epsilon = 0.01$).

Figure 8 depicts the relative weight of the triangulated Markov networks that are obtained after the decomposition by means of Algorithm 2. For a number of experiments, there is no decrease in network weight due to the decomposition, but for many experiments, network weights have become

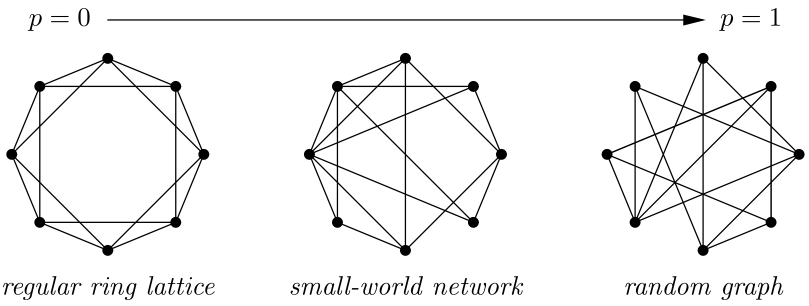

Figure 7: In order to construct a small-world graph, we start with a regular ring lattice ($N = 8$ and $K = 4$), and reconnect edges with probability p. At intermediate values of p, we obtain small-world networks. At high values of p, the small-world networks transform into random graphs.

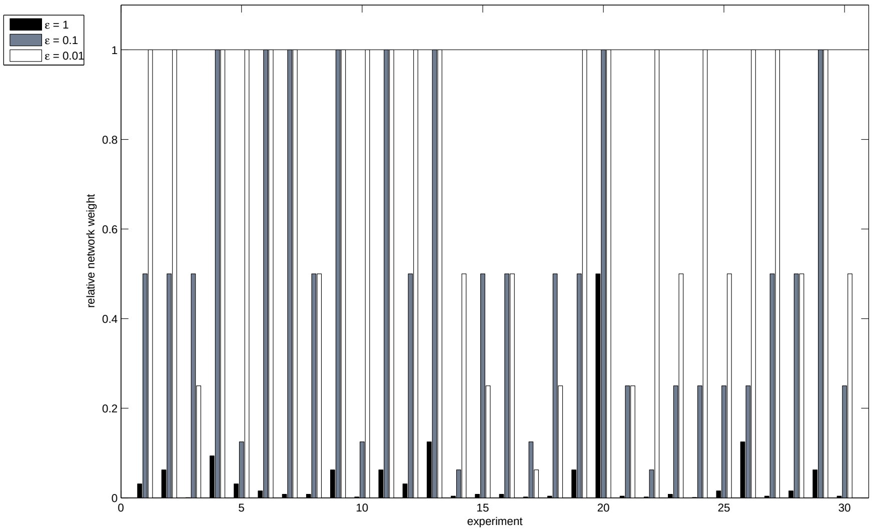

Figure 8: Relative weight of the triangulated and decomposed Markov networks as compared with the triangulated and non-decomposed Markov networks.

significantly smaller; showing that our algorithm performs well for these randomly generated Markov networks. For those Markov networks for which the network weight is decreased, the improvement will disappear as ϵ goes to zero, since an increasing number of components K will be chosen, unless we find a perfect approximation. Sometimes, the network weight becomes less for more accurate approximations (as in experiments 3,15,17, and 18). This is most likely caused by the way our greedy algorithm operates; for

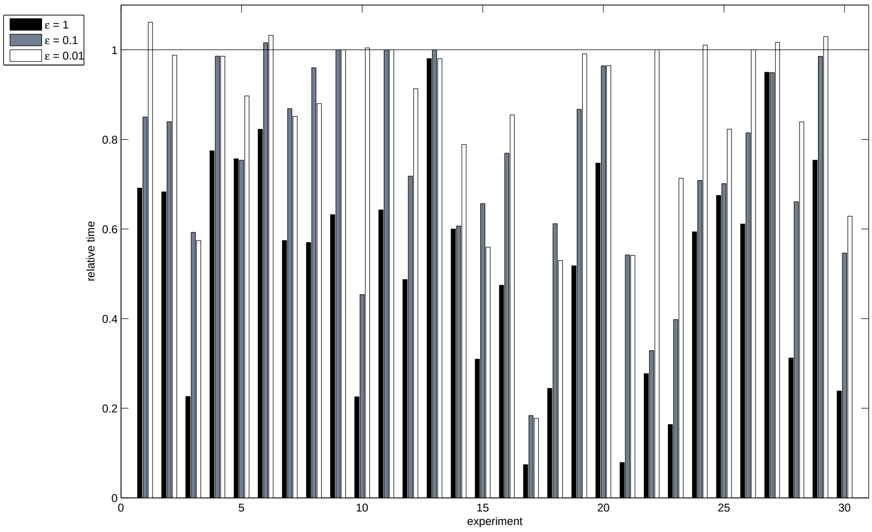

Figure 9: Relative inference times of the triangulated and decomposed Markov networks as compared with the triangulated non-decomposed Markov networks; decreases in relative network weight are not exactly matched by decreases in inference time.

a more accurate approximation, a set of different cliques may be selected for decomposition, which may perform better in terms of reducing network weight.

Smaller network weights do not always lead to corresponding decreases in inference time. Figure 9 depicts the inference time that is needed to compute the marginals for all fifty nodes. The shortest relative inference time is found for the Markov network of experiment seventeen with $\epsilon = 1$, where the time taken to compute the marginals with the decomposed Markov network is approximately 7% of the time taken to compute the marginals with the non-decomposed Markov network. The quality of the approximation of the marginals is of course dependent on ϵ, and Fig 10 depicts the average and standard deviation of the error in the marginals. It demonstrates that the approximation becomes better for decreasing values of ϵ.

Our best result was obtained for the Markov network of experiment seventeen with $\epsilon = 0.01$. This Markov network saw a decrease in network weight from $8.4 \cdot 10^6$ to $5.2 \cdot 10^5$ with an average difference of 10^{-3} in the computed marginals. Due to the decomposition of three cliques, the size of the largest clique after decomposition has decreased from twenty-four to nine-

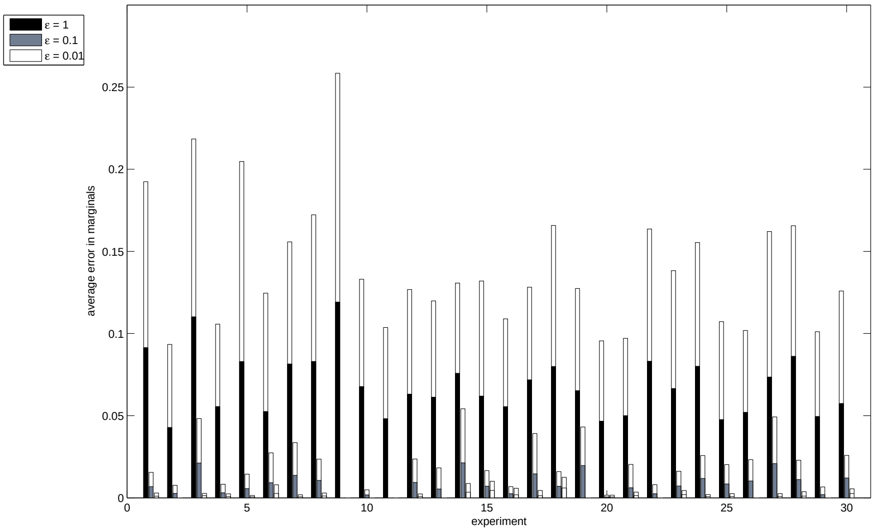

Figure 10: Average error in marginals when computed with the decomposed Markov network instead of the original Markov network, where stacked bars denote the standard deviation.

teen. Although the number of cliques after decomposition increased from twenty-seven to thirty-three, we did manage to compute the marginals in 18% of the time it would take without a decomposition since the complexity of inference scales with the largest clique size.

6 Discussion and Conclusion

The conducted experiments have shown that approximate inference by means of the rank-K approximation of potentials in a Markov network is feasible, where the trade-off between the efficiency of inference and the quality of the approximation is determined by the number of components K that is chosen. Note that the approximation that is given by Eq. (5) is only guaranteed to find the optimal rank-K approximation if the tensor $\mathcal{A}$ is orthogonally decomposable [25], which implies that there is room for further improvement. For instance, if a potential displays functional dependence, then a minimum number of components for an exact decomposition can be found analytically [18], whereas the rank-K approximation of Eq. (5) can only find better approximations to such potentials using an increasing number of components. Other decompositions, such as the (more general) Tucker decomposition [21], non-negative tensor decompositions [20], or decompositions in terms of decision trees [22], may also prove to be useful in this context.

The efficiency of the approximation also depends on the structure of the resulting junction tree, and in this paper, we have used a greedy approach to select those cliques that lead to minimal network weights. As we have seen, a decrease in network weight does not always imply the same decrease in inference time. The network weight also depends on the order in which the cliques are evaluated since different orderings suggest different cliques as candidates for decomposition; a more extensive search for the optimal candidates should improve the quality of the decomposition. Note that the decomposition can also be applied to Markov networks after triangulation, which may lead to further decreases in Markov network weight. Even though we may encounter problems such as those of Fig. 4, repeated decomposition could be useful for particular Markov networks. Especially Markov networks whose graphs represent small-world networks could benefit from such an approach, since these are characterized by large cliques that are sparsely connected.

The use of tensor decompositions for approximate inference is easily implemented and applied to arbitrary (discrete) Markov networks. Construction of the decomposed Markov network can be done off-line and standard inference algorithms may be used for approximate inference in the decomposed Markov network. The empirical validation of the algorithm that has been presented in this paper has shown the potential of this new approach to approximate inference in graphical models.

References

[1] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embedding in a k-tree. *SIAM J Alg Disc Meth*, 8(2):277–284, 1987.

[2] B. W. Bader and T. G. Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software*, 32(4):635–653, 2006.

[3] J. D. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35:283–319, 1970.

[4] L. de Lathauwer, B. de Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J Matrix Anal Appl*, 21:1253–1278, 2000.

[5] L. de Lathauwer, B. de Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J Matrix Anal Appl*, 21(4):1324–1342, 2000.

[6] F. J. Díez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain, 2006.

[7] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, London, UK, 1 edition, 1995.

[8] R. A. Harshman. Foundations of the PARAFAC procedure: Model and conditions for an "explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[9] J. Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11:644–654, 1990.

[10] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, UK, 1999.

[11] U. Kjaerulff. Triangulation of graphs - algorithms giving small total state space. Technical Report R-90-09, University of Aalborg, Aalborg, Denmark, 1990.

[12] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J Roy Stat Soc B*, 50:157–224, 1988.

[13] A. L. Madsen and F. V. Jensen. LAZY propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113(1–2):203–245, 1999.

[14] S. Milgram. The small world problem. *Psychol Today*, 2:60–67, 1967.

[15] K. P. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33, 2001.

[16] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In K. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, San Francisco, CA, 1999. Morgan Kaufmann.

[17] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, CA, 2nd edition, 1988.

[18] P. Savický and J. Vomlel. Tensor rank-one decomposition of probability tables. Technical Report DAR-UTIA 2005/26, Institute of Information Theory and Automation, Prague, Czech Republic, 2005.

[19] G. R. Shafer and P. P. Shenoy. Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2:327–352, 1990.

[20] A. Shashua and T. Hazan. Non-negative tensor decompositions with applications to statistics and computer vision. In *Proceedings of the 22nd International Conference on Machine Learning*, volume 119 of *ACM International Conference Proceeding Series*, pages 792–799, New York, NY, 2005. ACM Press.

[21] L. R. Tucker. Some mathematical notes of three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[22] M. A. J. van Gerven. Efficient Bayesian inference by factorizing conditional probability distributions. Technical Report ICIS-R6032, Radboud University, Nijmegen, The Netherlands, 2006.

[23] H. Wang and N. Ahuja. Compact representation of multidimensional data using tensor rank-one decomposition. In *International Conference on Pattern Recognition*, pages 44–47. IEEE, 2004.

[24] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[25] T. Zhang and G. Golub. Rank-one approximation to high order tensors. *SIAM J Matrix Anal Appl*, 23:534–550, 2001.