International Journal of Approximate Reasoning 109 (2019) 19–41

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Compact structures for continuous time Bayesian networks

Logan Perreault *, John Sheppard

Computer Science Department, Montana State University, Bozeman, MT 59715, USA

ARTICLE INFO

Article history:
Received 13 December 2018
Received in revised form 4 March 2019
Accepted 16 March 2019
Available online 21 March 2019

Keywords:
Continuous time Bayesian networks
Context
Structured conditional intensity matrix
Hierarchical clustering
Trees

ABSTRACT

The continuous time Bayesian network (CTBN) is a model capable of describing the probability distribution over a set of variables as it changes in time. The model relies on a directed graph structure to describe direct dependencies between variables, thereby simplifying the underlying parameters that describe the initial distribution and transition behavior. Although this approach can be effective in managing complexity, the size of the model can still be intractable if one variable depends directly on many other variables. To address this issue, we present two methods for imposing additional structure on the model that are capable of capturing regularities in the data that cannot be represented using the CTBN graph structure alone. We demonstrate how these methods can reduce model complexity and compare the representational capabilities of both approaches.

1. Introduction

Continuous time Markov processes (CTMPs) are mathematical models capable of representing discrete-state continuous-time systems of variables. Although these models are well-suited for representing the temporal behavior of small systems, complexity is exponential in the number of variables, making representation and inference over larger systems infeasible. In an attempt to address this concern, the continuous time Bayesian network (CTBN) was proposed as a more compact representation [17].

CTBNs take advantage of structural similarities in a CTMP and encode this information using a directed graph. Each node in the graph represents a variable in the underlying continuous time Markov process, while edges represent direct dependencies between these variables. Each node is parameterized separately, and the number of required parameters for each node is exponential in the size of the parent set for that node.

Although the compact representation afforded by the CTBN is a substantial improvement over traditional CTMPs, there are still scalability issues that must be addressed. The exponential complexity with respect to a node's parents may be problematic for large or highly inter-dependent systems. In the worst case, a node may depend on all other nodes in the network, resulting in a complexity that is equivalent to the original CTMP. For cases like these, additional compact representations must be employed to manage model complexity further.

The remainder of this paper is organized as follows. Background information is provided in Section 2, related work in Section 3, and the distance metrics we use are detailed in Section 4. We discuss clustering of intensity matrices in Sections 5 and 6. Mapped conditional intensity matrices are described in detail in Sections 7 and 8 and employ compositions of functions to encode similarities in a node's parameters. Sections 9 and 10 describe tree-structured conditional intensity matrices, which provide an alternative means of compactly representing parameters using decision trees. We wrap up by

* Corresponding author.
E-mail addresses: logan.perreault@msu.montana.edu (L. Perreault), john.sheppard@montana.edu (J. Sheppard).

https://doi.org/10.1016/j.ijar.2019.03.005
0888-613X/

providing a comparison of the representations in Section 11 and conclude with parting thoughts in Section 12. Differences between the representational capabilities of the each approach are also discussed.

2. Background

A continuous time Markov process (CTMP) is a model that describes the distribution over a continuous time random process, which is a random variable X, whose state evolutions are defined as a function of time. Unlike other temporal models like the dynamic Bayesian network [5], a CTMP explicitly represents time as a continuous random variable, avoiding the need to specify a particular time granularity. A CTMP over a process X consists of two parts: an initial distribution $P_X(0)$ and a transition intensity matrix $\mathbf{Q}_X$, each defined over the states of X. We use $|X|$ to denote the total number of states variable X can take. Each entry $q_{i,j}$ in row i, column j of the matrix $\mathbf{Q}_X$ defines the non-negative intensity with which the process will transition from state x_i to state x_j as a function of time. The diagonal entry for some row i column i is denoted $q_{i,i}$ or simply q_i, and is constrained to be the negative sum of the remaining row. Formally, $q_{i,i} = -\sum_{j \neq i} q_{i,j}$, which ensures that each row sums to zero, while describing the transition rate from state x_i to any other state x_j. The resulting model can be used to answer queries regarding the probability distribution over the states of a process over time.

The primary concern in specifying a CTMP is that the number of parameters that are required is exponential with respect to the number of variables in the process. Assume a system with multiple discrete random variables $\mathbf{X}$. A CTMP would represent this as a single discrete variable Y, where each state $y \in Y$ corresponds to a full state instantiation of each variable $X \in \mathbf{X}$. This makes the model unsuitable for even moderately complex systems of variables. To address this issue, the continuous time Bayesian network (CTBN) imposes structure on the CTMP. A CTBN is represented using a directed graph, where each node in the graph is a discrete variable $X \in \mathbf{X}$, and edges indicate conditional dependencies between variables. Rather than specifying a single CTMP that is intractable in size, CTBNs instead describe the system as a series of conditional CTMPs, one for each variable in the process. More specifically, the transition behavior for each variable is conditioned on the current state of the variables it directly depends on, which corresponds to its parents in the graph. To express this transition behavior, CTMPs use what is referred to as a conditional intensity matrix (CIM), which defines the intensities as a function of the parent states.

By defining a conditional CTMP for each variable in the system, the size of the parameterized model can be reduced greatly. The number of entries in an intensity matrix for a CTMP is

$$n_{ctmp} = \left(\prod_{X \in \mathbf{X}} |X| \right)^2.$$

Alternatively, the total number of entries in the CIMs of a CTBN over the same set of variables is

$$n_{ctbn} = \sum_{X \in \mathbf{X}} \left(\left(\prod_{U \in \mathbf{Pa}(X)} |U| \right) \times |X|^2 \right), \tag{1}$$

where $\mathbf{Pa}(X)$ is the set of parents for node X in the graph. This is because intensity matrices of size $|X|^2$ are defined over the local state space for each individual variable, and the number of intensity matrices that must be specified per variable is determined by the total number of state instantiations to its parents. If these parent sets are smaller than the total number of nodes in the graph, then $n_{ctbn} < n_{ctmp}$. In other words, so long as the behavior of a variable depends only on a subset of the other variables in the system, a CTBN is able to model the process more compactly.

Fig. 1 shows the graph structure of a CTBN that describes the influence of a drug on a patient [17]. This process consists of eight discrete variables: $\mathbf{X} = \{$`Concentration`, `Pain`, `Barometer`, `Drowsy`, `Uptake`, `Full Stomach`, `Hungry`, `Eating`$\}$. At a high level, the concentration of a drug in a patient's system is determined directly by the uptake of the drug and the contents of the patient's stomach. The drug itself influences the patient's pain and drowsiness. The contents of the patient's stomach is dependent on if they were eating, which is dependent on if they were hungry, which is dependent on if their stomach was full. Finally, a patient's pain level is not only influenced by the concentration of the drug, but also the barometric pressure.

In the drug effect network, a conditional Markov process is specified for each of the eight nodes in the network. As an example, consider the `Pain` node, which has parents `Barometer` and `Concentration`. For the sake of conciseness, these nodes can be referred to using their first initials P, B and C. Node P is parameterized using a CIM where the number of rows/columns is $|P|$. The CIM can be viewed as a set of homogeneous intensity matrices, one for each unique state instantiation of the parent nodes. The CIM for node P will therefore consist of $|B| \times |C|$ intensity matrices. If it is assumed that $|B| = 3$ and $|C| = 2$, then there are $3 \times 2 = 6$ homogeneous intensity matrices that make up the CIM for node P as shown below.

$$P_P = \begin{bmatrix} 0.1 & 0.9 \end{bmatrix}$$

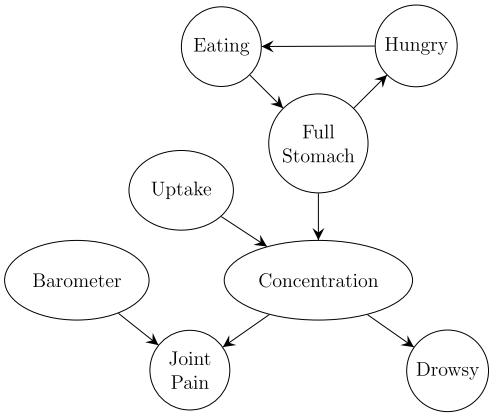


Fig. 1. Example drug effect network.

$$\mathbf{Q}_{P|B,C} = \left\{ \begin{array}{lll} \mathbf{Q}_{P|b_0,c_0} = \begin{pmatrix} -6 & 6 \\ 0.1 & -0.1 \end{pmatrix} & \mathbf{Q}_{P|b_0,c_1} = \begin{pmatrix} -1 & 1 \\ 0.1 & -0.1 \end{pmatrix} & \mathbf{Q}_{P|b_0,c_2} = \begin{pmatrix} -6 & 6 \\ 0.1 & -0.1 \end{pmatrix} \\ \mathbf{Q}_{P|b_1,c_0} = \begin{pmatrix} -1 & 1 \\ 0.3 & -0.3 \end{pmatrix} & \mathbf{Q}_{P|b_1,c_1} = \begin{pmatrix} -0.3 & 0.3 \\ 0.3 & -0.3 \end{pmatrix} & \mathbf{Q}_{P|b_1,c_2} = \begin{pmatrix} -1 & 1 \\ 0.3 & -0.3 \end{pmatrix} \\ \mathbf{Q}_{P|b_2,c_0} = \begin{pmatrix} -0.01 & 0.01 \\ 2 & -2 \end{pmatrix} & \mathbf{Q}_{P|b_2,c_1} = \begin{pmatrix} -0.01 & 0.01 \\ 2 & -2 \end{pmatrix} & \mathbf{Q}_{P|b_2,c_2} = \begin{pmatrix} -0.01 & 0.01 \\ 2 & -2 \end{pmatrix} \end{array} \right\}$$

Although the CTBN representation may be substantially more compact than a CTMP over the same set of variables, in many situations the model may still be unmanageably large. Note the product in Equation (1), which accounts for the requirement that all parent instantiations must be enumerated. If a node in a CTBN has a large number of parents, then storing and using a CIM for the node may be difficult due to the exponential number of homogeneous intensity matrices. In the worst case, all nodes may depend on every other node. In this case, no independencies can be exploited and a CTBN provides no benefits over a CTMP representation. For CTBNs with large parent sets, a more concise representation may be required. In this work, we introduce two such representations, both capable of reducing the complexity of specifying the parameters of a CTBN node when the number of parents is large.

3. Related work

The CTBN is not the only framework designed to induce structure on a CTMP; there is also the Kronecker representation and the decision-diagram [18]. In the former case, Kronecker algebra is able to decompose the intensity matrix of a CTMP into basic matrix operations [3]. These operations, known as Kronecker products and Kronecker sums, allow smaller matrices to be stored that can be used to reconstruct the original process after applying the Kronecker operations:

$$R = \sum_{e=1}^{E} \bigotimes_{l=1}^{L} R_e^{(l)}.$$

Here, R is a full joint rate matrix over L variables, $R_e^{(l)}$ is a rate matrix over variable l, and E can be viewed as the individual events in the process. Although the Kronecker representation strives toward a similar goal of compact representation, our approach starts with the already structured CTBN representation and seeks to provide additional structure to the conditional CTMPs. Our approach is also distinctly different from the operation decomposition used in Kronecker algebra.

In the latter case, Decision diagrams are used to encode functions over discrete domains compactly. This same concept can be used to encode intensity matrices, providing another way to represent CTMPs compactly [8]. Decision diagrams are represented as directed acyclic graphs where each layer corresponds to a different variable. The outgoing edges for each node in a layer corresponds to values that the variable can take on. Traditionally, decision diagrams are binary, but extensions have been made to allow for multiway decision diagrams (MDDs) [20]. The final value is determined by combining the values along the edges on a path from the root to a leaf node. This means that the behavior of a decision diagram largely depends on the operator that is used to combine edge values. Shelton and Ciardo discuss versions where the edge values are summed (EV$^+$MDD) and where the edge values are multiplied (EV*MDD) [18]. Here again, decision diagrams are used to add structure to a flat CTMP, while we look into further improving the CTBN framework.

The partitioned CTBN is another structured representation of a CTMP, and can be viewed as a generalization of the CTBN [25]. The concern being addressed here is that while complexity of unstructured CTMPs is unmanageable, there are assumptions made by the CTBN framework that may not suit certain data structures. Specifically, the conditional CTMP defined by a CTBN assumes that transition intensities only change as a function of the parent states. Weiss et al. lift this

restriction by allowing multiple intensities to be specified for each parent combination, and instead map intensities to partitions defined over the full joint of the variables in the model. These partitions are represented using trees or forests, which bears a loose resemblance to our work. The difference is that we use trees to encode a CIM more efficiently rather than replace the CIM with a different representation entirely. Furthermore, while Weiss et al. are addressing an issue of representation, we are attempting to improve scalability. The partition-based CTBN is actually a *more* complex model in general, requiring more parameters as well as a partitioning structure that is specified in addition to the original CTMP. In contrast, our approach adds structure to the CTBN rather than replacing the existing structure, and generally requires less parameters.

Boutilier et al. introduce the concept of context-specific independence in Bayesian networks [2]. In their work, they formalize scenarios where a variable is independent of the state of a subset of its parents. This allows for conditional probability tables to be encoded using a decision tree, reducing the total number of parameters required to specify the Bayesian network. The tree-structured CIMs presented in this work also make use of context-specific independence and are the continuous time analog of tree-structured conditional probability tables. As such, the work by Boutilier et al. serves as a static foundation to the techniques presented in Section 9. We not only extend these methods to work in continuous time, but Section 7 also introduces an alternative method for encoding CTBN parameters compactly that is unrelated to tree structures.

4. Distance metrics for transition distributions

Transitions between the states of a variable in a CTBN are described using exponential distributions. Let p_1 and p_2 be exponential distributions with rates λ and μ respectively. Furthermore, let $f(t)$ be the probability density function (PDF) for p_1, and let $F(t)$ be the cumulative distribution function (CDF) for p_1. Similarly, let $g(t)$ and $G(t)$ be the PDF and CDF for p_2. The distance between the two exponential distributions p_1 and p_2 can be quantified using a wide array of metrics, each of which has its own unique attributes and advantages [9]. For the purposes of this work, three common metrics used to compare probability distributions are considered. The section concludes with a brief description of how these distance metrics may be used to compare intensity matrices.

4.1. Symmetric Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence, also referred to as relative entropy, measures the divergence of an approximating distribution from a target distribution [13]. The notation $D_{KL}(p_1||p_2)$ is used to indicate the KL divergence of distribution p_2 from p_1, where p_2 is the approximating distribution, and p_1 is the target distribution. The following shows the derivation for KL divergence for the special case where p_1 and p_2 are exponential distributions as described earlier in this section. Note that because p_1 and p_2 are positive distributions, the integral starts at zero.

$$\begin{aligned}
D_{KL}(p_1||p_2) &= \int_0^\infty f(t)\log\left(\frac{f(t)}{g(t)}\right)dt \\
&= \int_0^\infty \lambda e^{-\lambda t}\log\left(\frac{\lambda e^{-\lambda t}}{\mu e^{-\mu t}}\right)dt \\
&= \left(\lambda(\mu-\lambda)\int_0^\infty te^{-\lambda t}dt\right) + \left(\lambda(\log\lambda-\log\mu)\int_0^\infty e^{-\lambda t}dt\right) \\
&= \left(\lambda(\mu-\lambda)(-\lambda^{-2}e^{-\lambda t}(\lambda t+1))\right) + \left(\lambda(\log\lambda-\log\mu)(-\lambda^{-1}e^{-\lambda t})\right)\Big|_0^\infty \\
&= \frac{\mu}{\lambda} + \log(\lambda) - \log(\mu) - 1
\end{aligned}$$

Although useful for many applications, KL divergence is an asymmetric measure, and is therefore unsuitable for a distance measure intended to test the similarity of between arbitrary distributions. There have been several different adaptations to KL divergence in the literature that extends the measure to enforce symmetry [15,1]. In this work, we use the definition that sums the standard KL divergence between the two probability distributions in both directions. This alternative metric is referred to as symmetric KL divergence, and is denoted $\widetilde{D}_{KL}(p_1, p_2)$:

$$\begin{aligned}
\widetilde{D}_{KL}(p_1, p_2) &= D_{KL}(p_1||p_2) + D_{KL}(p_2||p_1) \\
&= \left(\frac{\mu}{\lambda} + \log(\lambda) - \log(\mu) - 1\right) + \left(\frac{\lambda}{\mu} + \log(\mu) - \log(\lambda) - 1\right)
\end{aligned}$$

$$= \frac{\lambda^2 + \mu^2}{\lambda \cdot \mu} - 2.$$

4.2. Hellinger distance

The Hellinger distance has been used successfully to quantify the distance between two probability distributions [26,4]. Denoted $D_H(p_1, p_2)$, the Hellinger distance when considering exponential distributions is as follows.

$$\begin{aligned}
D_H(p_1, p_2) &= \frac{1}{2} \int_0^\infty \left(\sqrt{f(t)} - \sqrt{g(t)} \right)^2 dt \\
&= 1 - \int_0^\infty \sqrt{f(t)g(t)} dt \\
&= 1 - \int_0^\infty \sqrt{\lambda e^{-\lambda t} \mu e^{-\mu t}} dt \\
&= 1 - \sqrt{\lambda\mu} \int_0^\infty e^{-(\lambda+\mu)t/2} dt \\
&= 1 - \sqrt{\lambda\mu} \left(-\frac{2}{\lambda+\mu} e^{-(\lambda+\mu)t/2} \right) \Big|_0^\infty \\
&= 1 - \frac{2\sqrt{\lambda\mu}}{\lambda+\mu}
\end{aligned}$$

4.3. Kolmogorov metric

The Kolmogorov metric, also referred to as the uniform metric, is another means of measuring differences in probability distributions [12,27]. This metric indicates the largest deviation between the cumulative distribution functions and is denoted $D_K(p_1, p_2)$. The following definition shows the Kolmogorov metric when comparing exponential distributions:

$$\begin{aligned}
D_K(p_1, p_2) &= \sup_t |F(t) - G(t)| \\
&= \sup_t |(1 - e^{-\lambda t}) - (1 - e^{-\mu t})| \\
&= \sup_t |e^{-\mu t} - e^{-\lambda t}|
\end{aligned}$$

where $\sup_t$ is the supremum over the domain of t. To identify the maximum value, the terms within the absolute value function can be derived.

$$\frac{d}{dt} e^{-\mu t} - e^{-\lambda t} = \lambda e^{-\lambda t} - \mu e^{-\mu t}$$

By setting this equal to zero and solving for t, it is possible to determine the time at which the maximal difference occurs.

$$t = \frac{\log\lambda - \log\mu}{\lambda - \mu}$$

This value may then be substituted back into the original equation to obtain the supremum, and therefore the distances according to the Kolmogorov metric.

$$\begin{aligned}
D_K(p_1, p_2) &= \left| e^{-\mu((\log\lambda - \log\mu)/(\lambda-\mu))} - e^{-\lambda((\log\lambda - \log\mu)/(\lambda-\mu))} \right| \\
&= \left| \left(\frac{\lambda}{\mu}\right)^{\mu/(\mu-\lambda)} - \left(\frac{\lambda}{\mu}\right)^{\lambda/(\mu-\lambda)} \right|
\end{aligned}$$

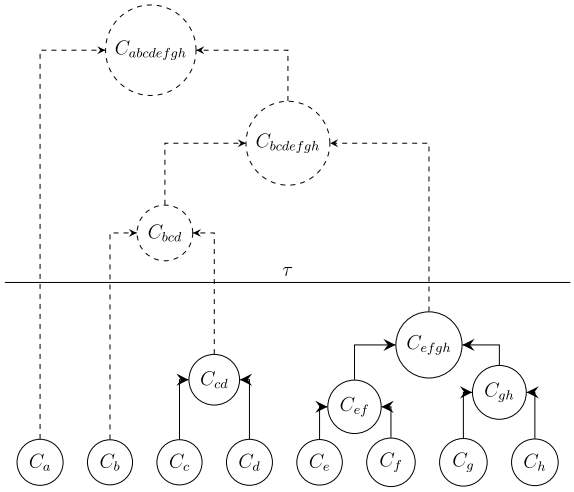


Fig. 2. Intensity matrix clustering.

4.4. Applying distance metrics to intensity matrices

To compare intensity matrices, the distance metrics described previously in this section can be leveraged. Each rate $q_{i,j}$ in an intensity matrix Q specifies a potentially unique exponential distribution p_1, which can be compared to the corresponding exponential distribution p_2 defined by rate $q'_{i,j}$ in a matrix Q'.

Let $D(q_{i,j}, q'_{i,j})$ be the distance between the exponential distributions defined by rates $q_{i,j} \in Q$ and $q'_{i,j} \in Q'$. Then let $D(Q, Q') = \sum_{i,j} D(q_{i,j}, q'_{i,j})$. Each of the distances $D(q_{i,j}, q'_{i,j})$ may be computed using one of the metrics discussed in this section. Note that the assumption of symmetry holds at the matrix level as well, such that $D(Q, Q') = D(Q', Q)$.

5. Hierarchical clustering of intensity matrices

The basic idea behind representing a CIM compactly is that similar intensity matrices that comprise the CIM may be consolidated. The task then becomes how best to group intensity matrices so as to eliminate the largest number of intensity matrices while still retaining as much of the original semantics encoded by the network as possible. Although it would be possible to identify a score for a particular grouping of intensity matrices based on the distance metrics discussed in Section 4, it is infeasible to consider every possible grouping for all but the smallest of CIMs. Instead, clustering techniques are employed to manage the complexity of identifying a suitable grouping of intensity matrices.

Although there are a variety of clustering algorithms available, this work makes use of hierarchical clustering [16]. Hierarchical clustering is either performed using an agglomerative or divisive approach. Agglomerative clustering works by starting with each datapoint in a separate cluster and iteratively merging the clusters that are closest to one another. Conversely, divisive clustering initially places all datapoints into a single cluster, and divides clusters into groups that are farthest away from one another. An agglomerative approach to clustering is used in this work to identify groups of similar intensity matrices. The clustering process depends on the ability to quantify the similarity between two arbitrary clusters. Cluster similarity can be achieved by making use of a linkage criterion that indicates distance between clusters based on pairwise distances between elements within the clusters. In this context, the elements within clusters are intensity matrices, and determining the distance between these intensity matrices can be achieved by using the metrics discussed in Section 4. While a variety of linkage criteria have been defined in the literature, this work makes use of average linkage clustering. The criterion works by computing the distance between sets of clustered intensity matrices as the average of the pairwise distances between the elements in the sets and is a common choice of criterion when the underlying structure of the data is unknown [24].

Consider a node with parent variables $\mathbf{U}$ having eight possible state instantiations. The corresponding CIM therefore consists of eight intensity matrices, which are denoted: $\{\mathbf{Q}_a, \mathbf{Q}_b, \mathbf{Q}_c, \mathbf{Q}_d, \mathbf{Q}_e, \mathbf{Q}_f, \mathbf{Q}_g, \mathbf{Q}_h\}$. Clusters of these matrices are denoted $C_{\mathbf{u}}$, where $\mathbf{u}$ indicates the matrices contained in the cluster. For example, C_{be} is a cluster containing the matrices $\mathbf{Q}_b$ and $\mathbf{Q}_e$. Fig. 2 shows a notional example of how these matrices might be clustered hierarchically, using a dendrogram representation [19]. Here, the clustering starts at the bottom with each of the eight matrices assigned to individual clusters. Clusters are then combined in ascending order of distance, which is represented using the y-axis in the graph. For this example, it is determined that C_e and C_f are the closest to one another, and the two clusters are therefore combined into a single new cluster C_{ef}. This process is repeated continually until all clusters have been merged or until a stopping criterion is met.

Typically a stopping criterion either places a limit n on the depth of the tree or specifies a maximum distance threshold τ that determines whether or not to merge a pair of clusters. We make use of the distance threshold to determine which

Table 1
Example clustering of a CIM.

C_1:	$\mathbf{Q}_a$			
C_2:	$\mathbf{Q}_b$			
C_3:	$\mathbf{Q}_c$	$\mathbf{Q}_d$		
C_4:	$\mathbf{Q}_e$	$\mathbf{Q}_f$	$\mathbf{Q}_g$	$\mathbf{Q}_h$

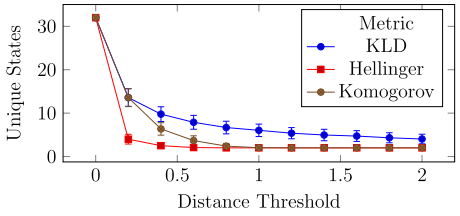


Fig. 3. Consistent distance metric experiment.

intensity matrices should be merged. Here, the distance threshold can be used as an approximation parameter, where smaller thresholds indicate more accurate approximations with smaller cluster sizes, and a distance threshold of zero will combine only intensity matrices that are exactly equal to one another. The choice of the maximum distance threshold depends on the desired approximation accuracy, as well as the underlying distance metric that is used to compare intensity matrices.

The dashed lines and nodes in Fig. 2 represent potential cluster merges that are not taken in this example because their distances exceed a specified threshold τ. The clusters on the frontier of the solid region in the hierarchy are C_a, C_b, C_{cd}, and C_{efgh}. Table 1 lists these four clusters in terms of the matrices they contain. Here, each row can be treated as a single intensity matrix since each entry in a row is identical or nearly identical to the rest. For instance, C_4 consists of four intensity matrices that can all be treated as equivalent to one another. By taking the mean of these intensity matrices, a single matrix $\mathbf{Q}_4$ may be used to represent the entire cluster. As a result, even though there are eight intensity matrices in this example, the entire set can be represented with only four unique intensity matrices that are obtained by computing the mean of each cluster.

6. Clustering experiments

In this section, an experiment is developed to compare the distance metrics derived in the previous section and determine their impact when clustering intensity matrices. This section focuses exclusively on the behavior of the clustering algorithm, and an investigation on the impact of the underlying CTBN dynamics is reserved for Sections 8 and 10. There are many ways of evaluating the quality of a clustering. Some evaluation techniques take a statistical approach, considering clusters with respect to the underlying data [10]. Another approach is to perform a relative comparison of the clusters, which is less expensive than performing statistical testing [11]. A traditional goal is to maximize inter-cluster distances while minimizing intra-cluster distances. For our purposes, we are interested in the total number of consolidated matrices and therefore treat the number of clusters as the variable of interest. For more specific applications where data is known to behave in a predictable fashion, distance metrics can be chosen based on more sophisticated criteria.

For this experiment, network structures were fixed to consist of five parent nodes and a single child node, each with two states. The networks were parameterized randomly such that each of the off-diagonal rates in the intensity matrices were drawn from a uniform distribution $U(0.0, 1.0)$. Diagonal entries are defined as the negative sum of the remaining row, as per the definition of a Markov process. To obtain a reasonable sample base, 100 different networks were generated in this fashion. For each network, the hierarchical clustering algorithm was run using each distance metric, with the maximal difference threshold ranging from 0.0 to 2.0. The results are shown in Fig. 3, where the total number of unique, unconsolidated matrices are shown on the y axis, and the maximal distance threshold is given by the x axis. The error bars show standard error for the 100 runs of the clustering algorithm.

The most important information that can be drawn from these results is the points at which each of the distance metrics tend to converge. For instance, the Hellinger metric appears to converge almost immediately, while KL divergence continues to decrease until nearly the end of the range. This provides an upper bound for a reasonable threshold value for each metric given the generated models. These ranges differ from metric to metric, meaning that the distance threshold should not necessarily be interpreted the same for each. As such it does not necessarily make sense to compare each metric's impact on the clustering algorithm using a fixed range of 0.0 to 2.0. To account for this, the experiment is rerun with the same setup, except that the approximation threshold for Hellinger is now varied from 0.0 to 0.5, Kolmogorov ranges from 0.0 to 0.75, and the threshold for symmetric KL divergence remains at the range 0.0 to 2.0. This normalizes each metric to a range that extends only to its observed convergence point where all clusters are merged. The plot for this modified version of the experiment is shown in Fig. 4, where the x axis now signifies the percentage into the threshold range for each metric. For instance, 0.5 indicates a threshold of 1.0 for symmetric KL divergence, but a threshold of 0.25 for Hellinger.

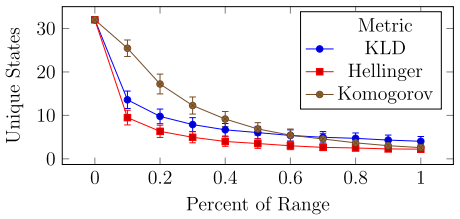


Fig. 4. Normalized distance metric experiment.

By changing the threshold ranges to suit each metric, the behavior of the clustering algorithm is depicted more clearly. Specifically, we are now able to observe the early stages of Kolmogorov and Hellinger-based clustering with finer granularity, now giving the appearance that Kolmogorov tapers off much more slowly than Fig. 3 implies. Based on this new plot, it is evident that symmetric KL divergence and Hellinger exhibit similar behavior in that cluster sizes increase rapidly with the size of the approximation threshold. Furthermore, although the number of clusters decreases rapidly with symmetric KL divergence, this behavior slows rather quickly, and by the end of the ranges considered it actually produces more clusters than either Hellinger or Kolmogorov. Finally, over the threshold interval from 0.0 to 0.75, a clustering algorithm using Kolmogorov combines intensity matrices at a more gradual rate. Based on the apparent convergence exhibited by the symmetric KL divergence metric that occurs prior to reaching a single cluster, we choose to use KL divergence going forward for the remainder of our experiments. Again, the choice of distance metric will generally depend on the underlying data, and future work in this area should consider distance metrics within the context of the problem being solved. Regardless of the metric used, a threshold value should be chosen based on the change in behavior of the underlying CTBN, as discussed in later sections.

7. Mapped conditional intensity matrices

The hierarchical clustering algorithm discussed in the previous section provides groups of intensity matrices that may be treated as equivalent Markov processes within a CIM. To exploit these equivalences, a computationally efficient data structure must be leveraged that reduces either the space requirements to store a CTBN or the time needed to run inference. Given the existing many-to-one relationship in the output of the clustering algorithm, a discrete mapping function is a natural choice to represent intensity matrix similarities. The basic idea is to consolidate similar intensity matrices by mapping their corresponding parent instantiations to a single output state. Let $\mathbf{u}$ be the set of unique instantiations for the parent variables $\mathbf{U}$. A discrete function f may be used to map inputs $\mathbf{u}$ to a set of output states $\mathbf{m}$, where $|\mathbf{m}| < |\mathbf{u}|$, and each element $m \in \mathbf{m}$ is mapped to an element $u \in \mathbf{u}$. In other words, the function f is surjective, and traditionally denoted as $f : \mathbf{u} \twoheadrightarrow \mathbf{m}$.

Referring back to the example clustering from Table 1, a discrete function f may be obtained simply by reversing the direction of the cluster notation.

$$\begin{aligned} \{\mathbf{a}\} &\rightarrow \mathbf{Q}_1 \\ \{\mathbf{b}\} &\rightarrow \mathbf{Q}_2 \\ \{\mathbf{c}, \mathbf{d}\} &\rightarrow \mathbf{Q}_3 \\ \{\mathbf{e}, \mathbf{f}, \mathbf{g}, \mathbf{h}\} &\rightarrow \mathbf{Q}_4 \end{aligned}$$

Recall that each element $\mathbf{a}, \mathbf{b}, \ldots, \mathbf{h}$ represents a unique state instantiation to the parent variables $\mathbf{u} \in \mathbf{U}$. Each of the output states $\mathbf{m} \in \mathbf{M}$ is an intensity matrix, defined to be the mean of the intensity matrices in the corresponding cluster. Using this mapping, it is possible to represent the example CIM with only four intensity matrices rather than the original eight. This new representation is referred to as a mapped CIM (MCIM).

To place this in a more explicit context, consider the example CTBN shown in Fig. 5 containing a single child and three parents. Notationally, a possible mapping for this network for a particular clustering could correspond to the following.

$$f(\mathbf{U}) = \begin{cases} \{a_0 b_0 c_0,\ a_0 b_1 c_1\} & \rightarrow \mathbf{Q}_1 \\ \{a_1 b_0 c_1,\ a_1 b_1 c_0,\ a_1 b_1 c_1\} & \rightarrow \mathbf{Q}_2 \\ \{a_0 b_0 c_1\} & \rightarrow \mathbf{Q}_3 \\ \{a_0 b_1 c_0\} & \rightarrow \mathbf{Q}_4 \\ \{a_1 b_0 c_0\} & \rightarrow \mathbf{Q}_5 \end{cases} \tag{2}$$

This mapping can be represented in two different ways. The first is that the standard CIM representation can be augmented to accommodate the new discrete mapping function f and will only specify an intensity matrix for each output state of the

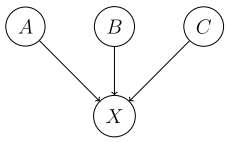


Fig. 5. Three-parent example network.

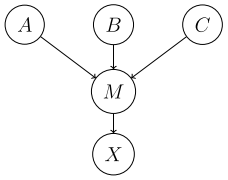


Fig. 6. Three-parent network after inserting mapping variable.

function. When an algorithm requires rates from an intensity matrix in a CIM, the intensity matrix is first looked up in the mapping function f based on the state instantiation of the parents.

The second option is to encode the mapping function f indirectly using standard CTBN semantics. This can be achieved by inserting a new synthetic variable, referred to as a mapping variable, into the network. In the three-parent example, the resulting structure after inserting the mapping variable is shown in Fig. 6. The mapping variable M has one state for every cluster, and is parameterized such that states are entered deterministically according to the parent instantiations. For example, consider the following matrix.

$$\mathbf{Q}_{M|a_1b_1c_0} = \begin{array}{c} \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{array} \begin{array}{c} \begin{array}{ccccc} m_1 & m_2 & m_3 & m_4 & m_5 \end{array} \\ \begin{pmatrix} -\infty & \infty & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \infty & -\infty & 0 & 0 \\ 0 & \infty & 0 & -\infty & 0 \\ 0 & \infty & 0 & 0 & -\infty \end{pmatrix} \end{array}$$

This process transitions to state m_2 instantaneously due to the infinite transition rates, regardless of the current state of the process. Upon entering state m_2, the process remains there indefinitely, until the parent instantiation changes and a different intensity matrix is employed. The process is forced into state m_2 because f maps the instantiation $a_1b_1c_0$ to matrix $\mathbf{Q}_2$. For conciseness, this type of deterministic matrix is denoted Q_s^n, where n is the number of states, and s is the state into which the process transitions. In the general case, the entries for these matrices are defined as follows.

$$q_{i,j} = \begin{cases} \infty, & \text{if } i \neq s \wedge j = s \\ -\infty, & \text{if } i \neq s \wedge j = i \\ 0, & \text{otherwise} \end{cases}$$

These deterministic intensity matrices are used to parameterize the CIM for the mapping variable to encode the deterministic function f. Finally the child node X in Fig. 6 is parameterized using the mean intensity matrices associated with each cluster. For instance, in the running example, $\mathbf{Q}_{X|m_2} = \mathbf{Q}_2$, as determined by the mapping function f.

Although the MCIMs discussed so far allow for the elimination of extraneous intensity matrices, there is a cost associated with representing the mapping between parent state instantiations and the corresponding matrices. Standard parameterization of a node X requires $n = \prod_{U \in \mathbf{Pa}(X)} |U|$ separate intensity matrices, and although a discrete function f may substantially reduce the number of required intensity matrices, the function itself must contain the n original parent state combinations in order to associate them with the new subset of intensity matrices. This problem is only exacerbated when the function is encoded using intensity matrices since the number of deterministic intensity matrices that are introduced is equivalent to the number of intensity matrices in the original parameterization. As an example, consider the network in Fig. 6, which actually has $|M|$ *more* intensity matrices than the original network from Fig. 5. Although this type of mapping has the advantage of improving interpretability by reducing the number of non-trivial parameters in the model, it does not reduce the overall complexity and may, in fact, have a negative impact. To combat this issue, this section discusses cases where discrete mappings can be decomposed to cover a smaller subset of parent variables, thereby reducing the space complexity.

To describe discrete mapping decomposition properly, some new terminology must be introduced. First, consider a subset $\mathbf{V}$ of the original parent set $\mathbf{U}$ for a node X. Going forward, this subset of variables is referred to as the *subject set*, and an instantiation $\mathbf{v}$ of the subject set is called a *subject*. Similarly, the remaining parents $\mathbf{C} = \{\mathbf{U} \backslash \mathbf{V}\}$ are referred to as the *context*

set, and an instantiation $\mathbf{c}$ is a *context*. The basic idea is that if the intensity matrices for two or more subjects $\mathbf{V}' \subseteq \mathbf{V}$ are equal across all contexts $\mathbf{c} \in \mathbf{C}$, then the matrices may be consolidated using a discrete function over the reduced domain of $\mathbf{V}$. In this case, the subjects $\mathbf{V}'$ are said to be contextually equivalent to one another. As before, matrix equivalence can be determined using a clustering algorithm, and the notion of equivalence may be relaxed to allow for approximate mappings.

Contextual equivalence is described further by way of example. Consider a node X with parent nodes $\mathbf{U} = \{A, B, C, D\}$, each having two states. A standard CIM for node X will consist of $2^4 = 16$ intensity matrices. An example of a discrete map $f_{\mathbf{U}}$ obtained by clustering the intensity matrices is shown below. Here, the total number of intensity matrices is reduced from 16 to 8, but the number of state instantiations that are stored for the domain of $f_{\mathbf{U}}$ is still 16.

$$f_{\mathbf{U}} = \begin{cases} \{a_0b_0c_0d_0,\ a_0b_1c_0d_0,\ a_1b_0c_0d_0\} & \rightarrow \mathbf{Q}_1 \\ \{a_0b_0c_0d_1,\ a_0b_1c_0d_1,\ a_1b_0c_0d_1\} & \rightarrow \mathbf{Q}_2 \\ \{a_0b_0c_1d_0,\ a_0b_1c_1d_0,\ a_1b_0c_1d_0\} & \rightarrow \mathbf{Q}_3 \\ \{a_0b_0c_1d_1,\ a_0b_1c_1d_1,\ a_1b_0c_1d_1\} & \rightarrow \mathbf{Q}_4 \\ \{a_1b_1c_0d_0\} & \rightarrow \mathbf{Q}_5 \\ \{a_1b_1c_0d_1\} & \rightarrow \mathbf{Q}_6 \\ \{a_1b_1c_1d_0\} & \rightarrow \mathbf{Q}_7 \\ \{a_1b_1c_1d_1\} & \rightarrow \mathbf{Q}_8 \end{cases}$$

To compact the representation further, the discrete function $f_{\mathbf{U}}$ can be decomposed by taking advantage of context-independent equivalence. In this case, $\mathbf{V} = \{A, B\}$ will be treated as the subject set, implying that $\mathbf{C} = \mathbf{U} \backslash \mathbf{V} = \{C, D\}$ is the context set. By inspecting the first four mappings in the discrete function $f_{\mathbf{U}}$, it can be seen that subjects a_0b_0, a_0b_1 and a_1b_0 are equivalent across all contexts. Note that each of the three subjects appear in each of the first four sets in $f_{\mathbf{U}}$, and the context does not change within each of these sets. The basic idea is that if the context is known, then there is no difference between the consolidated subject states. For instance, if the context is fixed to the state c_1d_0, then the transition behavior is described by the intensity matrix $\mathbf{Q}_3$ if the subject is equal to a_0b_0, a_0b_1 or a_1b_0, and by matrix $\mathbf{Q}_7$ if the subject is a_1b_1. This observed context equality can be used to derive two new discrete functions $f_{\mathbf{V}}$ and $f_{\mathbf{M}}$. The function $f_{\mathbf{V}}$ maps subject instantiations $\mathbf{v}_i$ to new states m_i. Function $f_{\mathbf{M}}$ then rewrites the original function $f_{\mathbf{U}}$ by replacing subject states with the corresponding state m_i obtained from function $f_{\mathbf{V}}$. These two functions are shown below.

$$f_{\mathbf{V}} = \begin{cases} \{a_0b_0,\ a_0b_1, a_1b_0\} & \rightarrow m_0 \\ \{a_1b_1\} & \rightarrow m_1 \end{cases}$$

$$f_{\mathbf{M}} = \begin{cases} \{m_0c_0d_0\} & \rightarrow \mathbf{Q}_1 \\ \{m_0c_0d_1\} & \rightarrow \mathbf{Q}_2 \\ \{m_0c_1d_0\} & \rightarrow \mathbf{Q}_3 \\ \{m_0c_1d_1\} & \rightarrow \mathbf{Q}_4 \\ \{m_1c_0d_0\} & \rightarrow \mathbf{Q}_5 \\ \{m_1c_0d_1\} & \rightarrow \mathbf{Q}_6 \\ \{m_1c_1d_0\} & \rightarrow \mathbf{Q}_7 \\ \{m_1c_1d_1\} & \rightarrow \mathbf{Q}_8 \end{cases}$$

Using function composition, the original function $f_{\mathbf{U}}$ may be rewritten in terms of $f_{\mathbf{M}}$ and $f_{\mathbf{V}}$, as follows.

$$f_{\mathbf{U}}(\mathbf{u}) = f_{\mathbf{M}}(f_{\mathbf{V}}(\mathbf{v}) \cup \mathbf{c}) \tag{3}$$

Here, the original state instantiation $\mathbf{u}$ passed as input to $f_{\mathbf{U}}$ is decomposed into the subject $\mathbf{v}$ and the context $\mathbf{c}$. The subject instantiation is then passed as input to the subject mapping function $f_{\mathbf{V}}$, producing an output m_i which is combined with the existing context instantiation $\mathbf{c}$. This newly obtained state instantiation is passed as input to the mapping function $f_{\mathbf{M}}$, which returns the correct intensity matrix.

Note that the domain of $f_{\mathbf{V}}$ is comprised of four state instantiations, and that $f_{\mathbf{M}}$ requires eight instantiations. In other words, there are 12 instantiations required to define both of the decomposed functions, compared to the 16 used for the original function $f_{\mathbf{U}}$. Furthermore, the instantiations for $f_{\mathbf{V}}$ and $f_{\mathbf{M}}$ cover two and three variables respectively, compared to the four variable instantiations in function $f_{\mathbf{U}}$. This decomposed representation encodes the exact same information as the original function $f_{\mathbf{U}}$, despite its more compact representation. Although a reduction from 16 instantiations to 12 may seem trivial, the advantages observed in this example are compounded as the size of the parent sets increase.

For a more concrete example, refer back to the drug effect network presented in Fig. 1. There exist identical intensity matrices in the CIM $\mathbf{Q}_{P|B,C}$ that can be consolidated using the principle of context-independent equivalence. For instance, if the barometric pressure B is known to be in state b_0, then that same transition behavior is expected regardless of whether

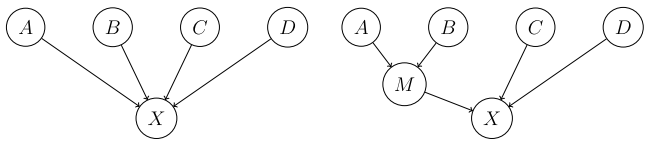


Fig. 7. Four-parent network before (left) and after (right) inserting a mapping variable based on a discrete function with context-independent equivalence.

the concentration of the drug C is in state c_0 or c_2. Similarly, when $B = b_1$, then there is no difference between c_0 and c_2. An even stronger example of context-independent equivalence is in the case where $B = b_2$, in which case all instantiations of C result in the same transition behavior. In other words, a patient's pain is fully independent of the concentration of the drug when the barometric pressure is known to be in state b_2.

It is worth noting that this decomposed mapping is a generalized version of the MCIM introduced earlier. Specifically, the mappings of the type shown in Equation (2) considered cases where the subject set was equal to the parent set ($\mathbf{U} = \mathbf{V}$), thus the context set was empty ($\mathbf{C} = (\mathbf{U} \backslash \mathbf{V}) = \emptyset$). This results in a mapping of $f_{\mathbf{U}}(\mathbf{u}) = f_{\mathbf{M}}(f_{\mathbf{V}}(\mathbf{u}))$, which can be represented more concisely using a single discrete function $f_{\mathbf{U}}$ relating inputs $\mathbf{u}$ to their corresponding intensity matrices.

As before, the decomposed functions can be encoded using standard CTBN semantics by introducing a new mapping variable. The key difference is that now that contextual equivalence has been introduced, the subject set $\mathbf{V}$ contains only a subset of the total number of parents $\mathbf{U}$. The mapping variable is parameterized according to the function $f_{\mathbf{V}}$; therefore, the parent set for the mapping variable is defined by $\mathbf{V}$. Similarly, the original child node X is reparameterized according to $f_{\mathbf{M}}$, meaning that the new parent set for X is $\{M \cup \mathbf{C}\}$. Fig. 7 shows a network before and after a new mapping variable M has been inserted to encode the discrete mapping. Note that the behavior of M is determined by nodes A and B, which are the members of the subject set. As before, M is parameterized using intensity matrices with deterministic transitions that map the parent instantiations to a new set of states $|M| < |A| \times |B|$. For this particular example, $|A| \times |B| = 4$, and $|M| = 2$. This new network requires $|M| \times |C| \times |D| = 8$ intensity matrices to parameterize X, and $|A| \times |B| = 4$ deterministic intensity matrices to parameterize M. These correspond to the domains for functions $f_{\mathbf{M}}$ and $f_{\mathbf{V}}$ respectively. This compact MCIM representation uses only twelve intensity matrices compared to the sixteen required by the original network. Once again, these savings become more pronounced as the size of the networks grow.

An additional benefit gained when taking advantage of context-independent equivalence is that the decomposed mapping functions can be inserted for multiple subject sets. For instance, multiple state combinations of variables A and B can be treated as equivalent for all contexts of C and D. Similarly, there may be state combinations of C and D that are equivalent regardless of the state of A and B. In this case, multiple mapping variables could be inserted, further decomposing the representation. The process of identifying proper mappings can be done efficiently in a pairwise fashion between variables, building up to potentially larger groups. This is analogous to the process of hierarchical clustering described earlier.

8. MCIM experiments

To evaluate MCIMs as a compact representation, experiments were conducted that compare CTBNs before and after introducing the mappings. For consistency, the discrete functions were represented using deterministically parameterized mapping nodes in each of the experiments. This section is broken into two parts, corresponding to unstructured and structured synthetic data. For the unstructured data experiments, models were produced by randomly generating each intensity matrix. Conversely, the structured data experiments generated intensity matrices using a pseudo-random procedure that followed specific constraints. The intent was to determine how structure in the data impacts the behavior exhibited by discrete mappings.

Throughout the experiments, the compact model was evaluated based on the change in complexity as well as the error introduced when compared to the original model. Since the compact models used in these experiments only changed the parameterization of a single node X, error was determined by querying the probability distribution over the states of X through time. Let G be a baseline model containing a node X, and let G' be a compacted model that simplifies the parameterization of node X. Let $P_{X(t)}$ and $P'_{X(t)}$ be the distributions over the states of variable X at time t obtained from running inference over G and G' respectively. Then error can be defined using discrete (nonsymmetric) KL-Divergence, shown below. Given that X is discrete, the KL-Divergence can be computed numerically by summing over the individual states. This quantifies the extent to which the compacted model G' diverges from the target baseline model G at some time t.

$$D_{KL}(P_{X(t)} || P'_{X(t)}) = \sum_{x \in X} P_{X(t)}[x] \log \frac{P_{X(t)}[x]}{P'_{X(t)}[x]}$$

This can be generalized to encompass an entire window of time $t = [t_s, t_e)$ by integrating over t.

$$D_{KL}(P_X || P'_X) = \int_{t_s}^{t_e} D_{KL}(P_{X(t)} || P'_{X(t)}) dt \tag{4}$$

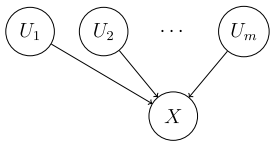


Fig. 8. Network structure for CTBNs used throughout experiments.

Note that while this is one possible error measure, other approaches may be better suited for specific applications. For instance, perturbation realization is a useful technique for evaluating parameter sensitivity in Markov processes, and has recently been adapted to work within the CTBN framework as well [23]. In essence, perturbation realization computes the expected long-term change in value of a function defined over a CTBN [21]. To identify long-term behavior, the steady-state distribution is identified. This process relies on the ergodicity assumption, which assures that no states are absorbing. For cases where this assumption can be guaranteed and interest lies in the long-term behavior of the process, perturbation realization could be used to evaluate the quality of the approximations discussed here. For the purposes of this work, however, we continue with the KL-Divergence error measure from Equation (4), which does not require that the models be ergodic and focuses on short-term behavior rather than using the steady-state distributions.

8.1. Unstructured data experiments

For the unstructured data experiments, networks were generated with a set of nodes $\mathbf{X}$. Consider a node $X_k \in \mathbf{X}$ with n_k states. The node X_k is parameterized using a CIM $\mathbf{Q}_{X_k|\mathbf{U}}$ with $n_k \times n_k$ dimensional intensity matrices. Each intensity matrix is generated randomly such that each off-diagonal entry $q_{i,j|i\neq j}$ is drawn from a uniform distribution $U(0.0, 1.0)$. Note that this is equivalent to sampling from a larger distribution range and observing a smaller time scale. After each of the $(n_k - 1) \times (n_k - 1)$ rates are obtained, the diagonal entries are set as the negated sum of the remaining row, as per the requirements for a valid intensity matrix. This process is repeated for each intensity matrix in the CIM $\mathbf{Q}_{X_j|\mathbf{u}} \in \mathbf{Q}_{X_j|\mathbf{U}}$. This, in turn, is applied to each node $X_j \in \mathbf{X}$.

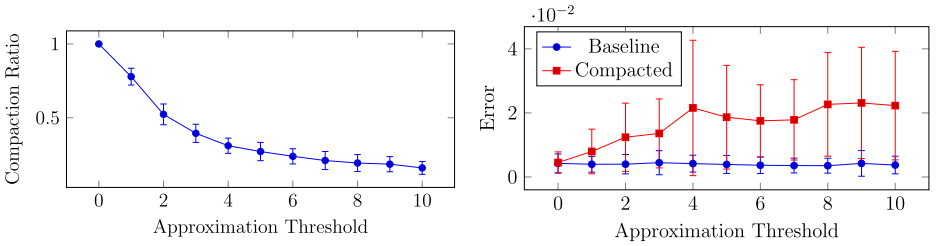
Given that the intent of the experiments was to demonstrate various CIM representations, the single-child multi-parent network structure used in previous examples was employed throughout the experiments to allow for a controlled means of changing the number of intensity matrices within the CIM. Formally, each network consisted of a single child variable X, as well as m parent variables U_i, where $i \in (1, m)$. A directed edge is added from each parent variable U_i to the child variable X. As a result, each of the m parents had a CIM consisting of a single intensity matrix with dimensions $|U_i| \times |U_i|$ states each, while the CIM for the child variable had $\prod_i |U_i|$ intensity matrices, each with dimensions $|X| \times |X|$. The described network structure is shown in Fig. 8. The number of states $|X|$, the total number of parent variables m, and the number of states $|U_i|$ for each parent were varied by experiment to investigate different aspects of compact CIM representations.

8.1.1. Approximation threshold

The first experiment was designed to investigate the effect of the approximation threshold in the clustering algorithm. The basic idea is that small approximation thresholds produce clusters that consist of intensity matrices that are very nearly identical to one another. Conversely, large thresholds consolidate more intensity matrices, thereby producing larger clusters with intensity matrices that are more dissimilar. To begin, a fixed network structure containing five binary parent nodes and a five-state child node was generated. Then, ten compact versions of the network were produced using an MCIM encoding obtained from clusters that were generated using a symmetric KL divergence measure with approximation thresholds varying from 0.0 to 10.0. Finally, inference was run over the compact network and the original network, with the KL divergence between the query results as shown in Equation (4) serving as an error measure. Importance sampling with 10000 samples was used as the inference algorithm [6,7]. To provide a baseline of comparison, error was measured against two inference runs over the same original network to quantify the error introduced by the approximate inference algorithm itself. This process was repeated 100 times to obtain a sufficiently large population of results. We define the compaction ratio metric as the ratio between the number of unique intensity matrices required by the MCIM representation and the original number of required intensity matrices. Let n be the number of intensity matrices required to specify a standard intensity matrix for some variable X, and let m be the number of matrices needed to parameterize the same variable using an MCIM. Then the compaction ratio $r = m/n$.

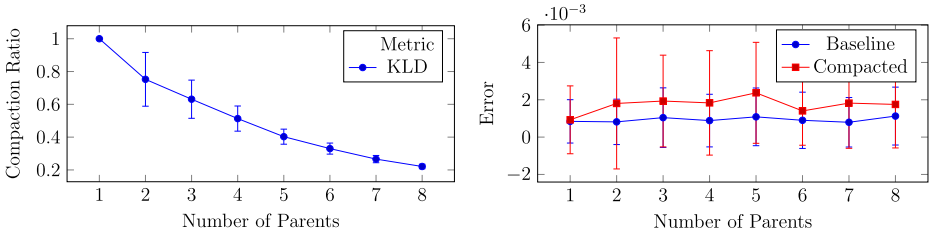
Fig. 9a shows the compaction ratio as a function of the approximation threshold, with error bars showing standard error. As such, smaller values indicate more efficient compact representations. The number of unique intensity matrices corresponds to the number of inputs to the function $f_{\mathbf{M}}$, or equivalently the number of intensity matrices for X in a CTBN with an inserted mapping variable M. Similarly, Fig. 9b shows the error for the compacted model, as well as the baseline error measure as a function of the approximation threshold, with the error bars representing the standard deviation of the KL-Divergence from the true distribution.

The behavior observed in Fig. 9a matches expectation. When the approximation threshold is set to zero, no compaction occurs, resulting in a compaction ratio of 1.0. Due to the random nature of the data, all produced clusters contain only a single intensity matrix, meaning that no consolidation occurs. As this approximation threshold is increased, cluster sizes



(a) Scalability analysis. (b) Accuracy analysis.

Fig. 9. Impact of approximation threshold on KL-Divergence error with structured data using a MCIM representation. Note the gradual upward trend for the compact representation in (b). The large error bars indicate the variance in approximation quality over 100 models, demonstrating that parameter configurations in the models ultimately dictate how well a compact representation will work.



(a) Scalability analysis. (b) Accuracy analysis.

Fig. 10. Impact of network structure with unstructured data using a MCIM representation.

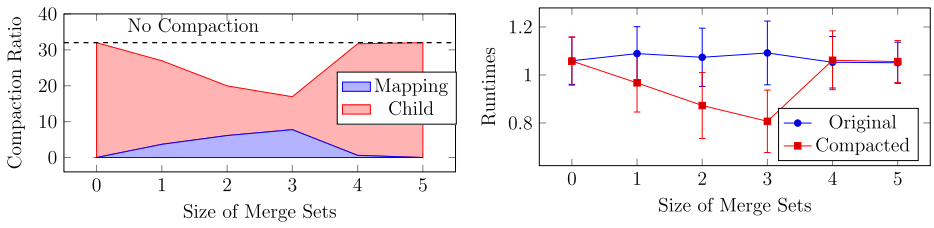
increase as well. These clusterings can be encoded using discrete mappings, which reduce the total number of unique intensity matrices. Note that a compaction ratio near zero indicates a small number of clusters. In the most extreme case, all intensity matrices may be merged into a single cluster, meaning that the child variable X acts approximately the same regardless of the states of its parents. If this were truly the case, the most efficient method for representing this independence would be to break all edges between the parents and the child and use a single intensity matrix to describe the now independent node's behavior. This is a process referred to as node isolation and can itself be used as an inference algorithm [22]. Given the random nature of the data, however, it is almost certainly the case that a large number of saved intensity matrices indicates an excessively large approximation threshold. Instead, a smaller approximation threshold should be employed that balances the accuracy of the approximation with the efficiency of the compact representation.

The error results shown in Fig. 9b indicate a steadily positive trend, where small approximation thresholds correspond to smaller error values, and large thresholds produce a larger error. This is not surprising in that large approximation thresholds allow increasingly diverse intensity matrices to be merged into a single cluster. Ultimately, this means that larger approximation thresholds will result in models that represent the same information with fewer parameters, which has the potential to introduce error. Another interpretation of the observed error behavior is that there is a trade-off between complexity and representational accuracy. A small approximation threshold will introduce very little error but may not provide much in the way of parameter reduction. Alternatively, a large approximation threshold will likely allow for substantial savings in the number of parameters but may introduce an excessive amount of error in the process.

8.1.2. Network structure

The next experiment was designed to determine the impact that the network graph structure has on the MCIM compact representation. Specifically, the structure from Fig. 8 was used where the number of parents m was varied from one to eight. Just as before, all parent nodes in the network were binary, while the child variable consisted of five states. For all intensity matrices, rates were drawn from uniform distributions ranging from 0.0 to 1.0. In this experiment, the symmetric KL divergence metric was used with an approximation threshold of 0.25 throughout. Models were compacted using the MCIM representation, and inference results were compared using the same error measure from the previous experiment. This process was performed 100 times, and the compaction ratio and error are plotted in Figs. 10a and 10b respectively.

As shown by Fig. 10a, the compaction ratio decreases with the size of the parent set. In other words, networks with more parents correspond to more consolidated intensity matrices. This increase in savings is due to the increase in the total number of intensity matrices. Networks are parameterized by drawing rates from a fixed uniform distribution, meaning that intensity matrices can be viewed as data points that fall within a fixed space of possible parameterizations. Given this random parameterization and the fixed approximation threshold, the density of the data is the primary factor influencing the results of the clustering algorithm. Since an increase in the number of parents produces an exponential increase in the number of intensity matrices, more of the space is covered with larger networks, allowing for more effective clusterings.



(a) Space analysis. (b) Time analysis.

Fig. 11. Impact of the size of merge sets with structured data using a MCIM representation. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Fig. 10b shows error as a function of the number of parents, both for the compacted model and the baseline. For networks with a single parent, the error appears to match the baseline, which makes sense given that there is no compaction occurring. For all other network sizes, the compacted model produces a consistently larger error than the baseline, likely due to the constant approximation threshold. The important aspect to note is that there is no discernible pattern for the compacted error in terms of either a positive or negative trend. This indicates that the number of parents does not have a significant impact on the error introduced by the MCIM representation. Furthermore, the selected approximation threshold in this case produces a consistently small error, and based on the error bars, the difference from the baseline is not statistically significant. This demonstrates that the MCIM representation becomes increasingly effective as the number of parents grows, with no degradation in error.

8.2. Structured data experiments

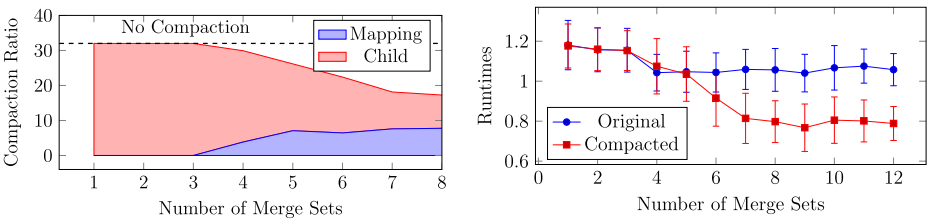
For the structured data experiments, networks were generated initially in the same fashion as the unstructured data experiments, with identical structure and random parameterization. The clustering algorithm used the symmetric KL divergence distance measure, with a constant approximation threshold of 0.25. Models were then modified to introduce structure synthetically into the data. Specifically, a subset $\mathbf{V}$ of parent variables was chosen randomly from the total set of parents $\mathbf{U}$. A set of state instantiations $\mathbf{v}'$ was then chosen randomly from the total set of instantiations $\mathbf{v}$ to $\mathbf{V}$. Each selected subset of state instantiations to the subset of parent variables is referred to as a merge set. The size of the merge sets as well as the total number of merge sets varied between experiments. The only stipulation was that the assignments be a valid subset such that $\mathbf{V} \subseteq \mathbf{U}$ and $\mathbf{v}' \subseteq \mathbf{v}$. These state instantiations were then merged such that the corresponding intensity matrices were assigned to be equivalent for each context, which was achieved by taking the mean of each group of $|\mathbf{v}'|$ matrices. To allow for approximations, noise was introduced to the rates in the intensity matrices. This was achieved by adding a random variable drawn from a uniform distribution $U(-\alpha, \alpha)$, producing intensity matrices that were approximately equal to one another. This procedure introduced context independence into otherwise unstructured data.

8.2.1. Merge set size

For the first structured experiment, the size of the merge sets was varied to determine the effectiveness of the compact representation. Specifically, the size of the sets ranged from zero to five, where zero indicates no data structure, and five means that state instantiations to all five parents are merged with no context. For each case, a maximum value of eight was set as the number of merge sets, meaning that at most eight states were altered to be approximately equivalent. Results are shown in Fig. 11.

Fig. 11a shows a stacked line graph of the size of the model, broken down by both intensity matrices and the discrete function. Specifically, the upper red region shows the number of unique intensity matrices that are stored for the child node, and the lower blue region shows the number of intensity matrices needed to parameterize an inserted mapping node to achieve the compact representation. The dashed line represents the size of the original model with no compaction applied. An important aspect to note is that the size of the model decreases when the number of merge sets increases to three, and increases again as it approaches five. This is because the reduction in the number of intensity matrices for the child node outweighs the increase in size that is introduced by inserting the mapping variable. This validates the intuition that the compact representation is most effective when the size of merge sets is somewhere between none and all of the parents.

Fig. 11b shows runtimes measured in seconds for inference run over both the original model shown as blue circles, and the compacted model shown by red squares, with error bars indicating standard error. As expected, the original model remains consistent regardless of the size of the merge sets. Runtimes for the compacted model decrease and increase proportional to the size of the model shown in Fig. 11a, and for the case where merge sets are of size three, the difference in runtimes is statistically significant. This decrease in runtime does not come from algorithmic improvements but instead occurs due to the reduction in the model size.



(a) Space analysis. (b) Time analysis.

Fig. 12. Impact of the number of merge sets with structured data using an MCIM representation.

8.2.2. Number of merge sets

The next structured experiment looks at the effect that the number of merge sets has on the resulting compact representation. The same network structure from the previous experiment was used, and in all cases the size of the merge sets was fixed to a size of three. The number of merge sets was varied from one to eight, where one indicates a model with no compaction, and eight is the total number of state instantiations for three binary parent variables, meaning that all state combinations for the parents are equivalent. The results are shown in Fig. 12.

Fig. 12a shows the size of the model, with the upper red region showing the number of intensity matrices required to parameterize the child node, and the lower blue region representing the number of intensity matrices introduced by inserted the mapping variable. Once again, the dashed line indicates the baseline number of intensity matrices in the original model prior to compaction. For very small numbers of merge sets, there is no compaction and the total number of intensity matrices matches the baseline. As the number of merge sets increases, discrete functions are introduced that ultimately decrease the total number of intensity matrices in the network, despite the need for parameterizing the new mapping variables.

Runtimes for this experiment are shown in Fig. 12b. Once again, there is no statistical difference between the runtimes for inference applied to the original model, regardless of the number of the merge sets. However, the runtimes for the compacted models decrease proportionally to the size of the model, which is reduced as the number of merge sets grows. When the number of merge sets is greater than six, the difference in runtimes is statistically significant. This shows that models may be represented more efficiently for structured data where the number of merge sets is large, both in terms of the space required to parameterize the model, and the time required to perform inference.

9. Tree-structured conditional intensity matrices

In addition to discrete mapping functions, a tree data structure may be used to represent a CIM compactly. This tree-structure representation of intensity matrices relates to context-specific independence in Bayesian networks, which uses a tree structure to store probabilities in a conditional probability table [2]. The intent of context-specific independence in Bayesian networks is to capture independencies that cannot be encoded using the graph structure alone, and the same principle holds in the domain of CTBNs. Boutilier et al. take advantage of regularities in the conditional probability table and encode these regularities using a tree structure. This section demonstrates how the same process can be applied to CIM regularities obtained via a clustering algorithm. The tree encoding of a CIM is referred to as a tree-structured CIM (TCIM).

Let X be a node in a CTBN with parents $\mathbf{U}$, and let $\mathbf{C}$ be the set of clusters obtained by hierarchically clustering the intensity matrices in the CIM for X. A tree T_X may be used to encode continuous-time context-specific independence exhibited by the clusters $\mathbf{C}$. Let $\mathbf{I}(T_X)$ be the set of interior nodes in tree T_X, and let $\mathbf{L}(T_X)$ be the set of leaf nodes. Furthermore, let $\mathbf{P}(T_X)$ be the set of paths in the tree, where each path $P \in \mathbf{P}(T_X)$ is a sequence of nodes from the root N_r to a leaf node $N \in \mathbf{L}(T_X)$. A variable U from the parent set $\mathbf{U}$ is assigned to each interior node $N \in \mathbf{I}(T_X)$, such that no path $P \in \mathbf{P}(T_X)$ contains more than one assignment $\mathbf{u}$. There are $|\mathbf{U}|$ edges leaving an interior node with assignment $\mathbf{u}$, each labeled with the states associated with variable U. Each leaf node $N \in \mathbf{L}(T_X)$ is assigned a unique intensity matrix. To retrieve an intensity matrix from T_X associated with the parent set instantiation $\mathbf{u}$, the tree is traversed starting at the root, following edges corresponding to the state assignments in $\mathbf{u}$, until an intensity matrix at a leaf node is reached.

The primary advantage of the TCIM representation is that entire branches of a tree may be pruned if all leaves in the subtree contain equivalent or approximately equivalent intensity matrices. In that case, the root of the subtree is replaced with a mean intensity matrix, creating a leaf that occurs at a higher level in the tree. Therefore, the paths of a pruned tree are potentially shorter than the total number of parents $|\mathbf{U}|$. This pruning process is how the TCIM structure achieves its compact representation.

An example is now presented to demonstrate how a TCIM may be used to store intensity matrices in a CIM efficiently. Consider a four-node network with a variable X and three parent variables $\mathbf{U} = \{A, B, C\}$. Here, $|A| = 3$, $|B| = 2$, and $|C| = 2$, for a total of twelve intensity matrices in the unmodified CIM $\mathbf{Q}_{X|\mathbf{U}}$. An example clustering of these intensity matrices is as follows.

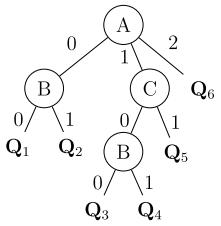


Fig. 13. Tree-structured CIM for node X.

C_1:	$\mathbf{Q}_{X\|\{a_0,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_0,c_1\}}$		
C_2:	$\mathbf{Q}_{X\|\{a_0,b_1,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_1\}}$		
C_3:	$\mathbf{Q}_{X\|\{a_1,b_0,c_0\}}$			
C_4:	$\mathbf{Q}_{X\|\{a_1,b_1,c_0\}}$			
C_5:	$\mathbf{Q}_{X\|\{a_1,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_1,b_1,c_1\}}$		
C_6:	$\mathbf{Q}_{X\|\{a_2,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_2,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_2,b_1,c_0\}}$	$\mathbf{Q}_{X\|\{a_2,b_1,c_1\}}$

Take, for example, cluster C_1. Both matrices in this cluster are tied to states a_0 and b_0, while the value of C varies across its two states. In other words, once it has been determined that $A = a_0$ and $B = b_0$, then it does not matter what the state of C is. More formally, X is contextually independent of C given a_0 and b_0. This contextual independence can be represented using a tree, as shown in Fig. 13. Note that if node A is followed down edge 0, and B is followed down edge 0, then the tree terminates at a leaf node with a value of $\mathbf{Q}_1$, which is the intensity matrix associated with cluster C_1. Following this path did not require evaluation of the variable C. Similar tree pruning is performed for the other clusters, and indeed there are a total of six distinct intensity matrices at the leaves of the tree, corresponding to each of the six clusters.

A number of algorithms exist that attempt to learn an optimal decision tree structure for classifiers, including CART, CHAID, ID3, and C4.5 [14]. In this work, we employ a greedy approach roughly based on CART, as shown in Algorithm 1. The algorithm begins by checking the set of clusters, and returns a representative intensity matrix if the tree has reached a leaf and there is only one cluster remaining. The algorithm then evaluates a score for each possible parent variable $U \in \mathbf{U}$, which is defined as the total number of clusters that contain each of the states for that variable. For example, at the top level of the tree in our previous cluster example, a_0 appears in 2 clusters, a_1 appears in 3 clusters, and a_2 appears in 1 cluster for a total score of 6, which is lower than the score for B or C. The variable with the lowest score is selected for inclusion in the decision tree, and the process is repeated for each possible child of the selected node, which will either consist of another variable split or a leaf node with a stored intensity matrix.

Algorithm 1 Greedily builds a tree-structured CIM.

Require: A node X with parent set $\mathbf{U}$, and a set of clusters $\mathbf{C}$ containing IMs $\mathbf{Q}_{X|U}$.

```
function BuildTree(U, C)
    if |C| == 1 then
        return Mean(C)
    end if

    minscore ← ∞
    M ← null
    for U ∈ U do
        U_score ← 0
        for i ∈ [0, |U|) do
            U_score ← U_score + CountClusters(C, U, u_i)
        end for
        if U_score < minscore then
            minscore ← U_score
            M ← U
        end if
    end for

    for i ∈ [0, |M|) do
        child ← BuildTree({U\M}, Filter(C, M, m_i))
        AddChild(M, m_i, child)
    end for

    return M
end function
```

Just as with the discrete functions described previously, the tree structures used to encode a CIM may be represented using standard CTBN semantics. In the Bayesian network literature, the analogous process is referred to as structural decomposition. This is achieved by retaining the variable associated with the root node of the tree as a parent of X, but removing

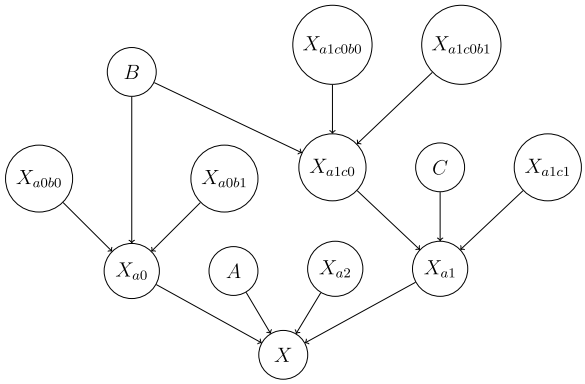


Fig. 14. Structure decomposition using the tree-structured CIM from Fig. 13.

all other variables. Additional variables are then inserted into the parent set of X, one for each state of the root variable, indicating the presence or absence of the state. These nodes in turn have parents associated with the next variable down the tree branch, along with all of the potential state instantiations associated with the variable.

Fig. 14 shows the structural decomposition for the tree in Fig. 13. First, note that nodes A, B, and C are the same nodes from the original network in Fig. 5 with identical parameterization. Conceptually, the newly introduced nodes can be thought of as random variables indicating the value of X given a specific context. For example, X_{a0} encodes the value that X takes on in the event that $A = a_0$. Nodes X_{a0}, X_{a1}, and X_{a2} can be obtained independently of one another, and the "correct" value can be retrieved after the value of A becomes known.

Nodes X, X_{a0}, X_{a1}, and X_{a1c0} are parameterized using deterministic CIMs that force the node to one of the parent Boolean values. As an example, if B is found to be b_0, then X_{a0} will transition with deterministic rates to ensure X_{a0} matches the state of X_{a0b0} at any point in time. Nodes X_{a2}, X_{a0b0}, X_{a0b1}, X_{a1c1}, X_{a1c0b0}, and X_{a1c0b1} are parameterized each with a unique intensity matrix corresponding to the six means of the clusters discussed previously. For instance, X_{a2} is parameterized using the intensity matrix corresponding to the mean of the matrices from cluster C_6 shown earlier in this section.

Again, consider the CIM $\mathbf{Q}_{P|B,C}$ in the example drug effect network for a more concrete example of continuous-time context-specific independence. When the barometric pressure B is known to be in state b_2, then the state of the drug concentration C is irrelevant. Using a tree representation, if B is the root of the tree and path b_2 is taken, then an intensity matrix can be stored at this level without the need to specify the value of C. Note that although there are two other pairs of identical matrices in $\mathbf{Q}_{P|B,C}$, they cannot be accounted for by continuous-time context-specific independence.

One concern when using standard CTBN semantics to represent a TCIM is the rapid increase in the number of nodes. Indeed, in many cases there is an increase in the total number of intensity matrices required to parameterize the network after introducing continuous-time context-specific independence. However, just as with the discrete mapping, there are additional advantages beyond parameter reduction. In terms of interpretability, the structure shown in Fig. 14 provides additional information about independence that was not immediately observable in the original network structure. In terms of efficiency, the new representation may reduce the maximal parent set size. Despite the increase in the total number of nodes, the size of the parent sets may be the more important factor when using inference algorithms that employ cluster graphs.

10. TCIM experiments

This section presents the results of several experiments designed to investigate the properties exhibited by TCIM representations. The experiments were structured in much the same way as the MCIM experiments from Section 8. Specifically, a TCIM representation was applied to unstructured data in Section 10.1, and structured data in Section 10.2. These experiments used the same network structure shown in Fig. 8, and the KL-Divergence error measure presented in Equation (4).

10.1. Unstructured data experiments

The TCIM unstructured data experiments were set up similarly to the MCIM unstructured experiments from Section 8.1. Each node in the network was parameterized using randomly generated intensity matrices. Specifically, each intensity matrix was produced by drawing off-diagonal transition rates from a uniform distribution $U(0.0, 1.0)$, with the diagonal entries in the matrix being implicitly defined by the remaining rates in the row. This type of network parameterization ensures that there are no dependencies between the intensity matrices, guaranteeing that there is no inherent underlying structure in the data.

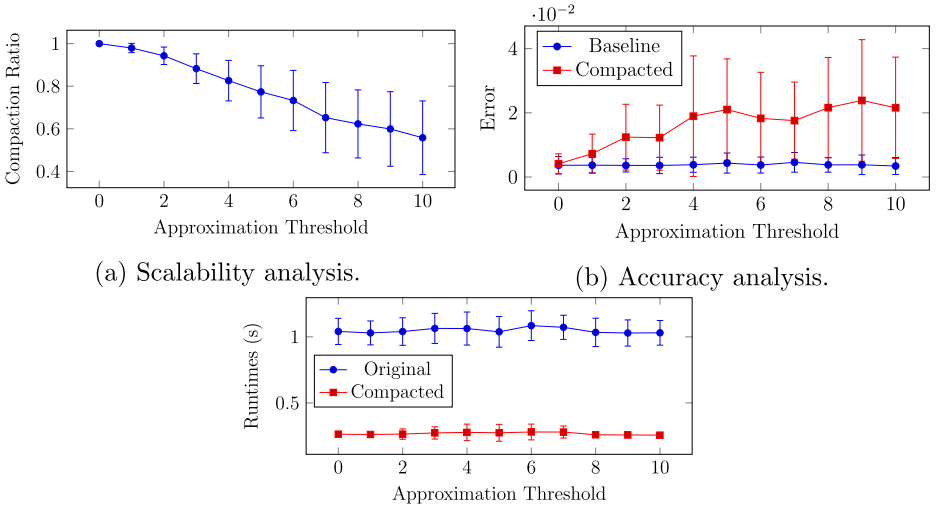


Fig. 15. Impact of approximation threshold with unstructured data using a TCIM representation. Note that runtimes remain constant at a mean of 1048 ms for the original, and 268 ms for the compact version.

10.1.1. Approximation threshold

The first experiment conducted with unstructured data focused on determining the impact of the clustering approximation threshold. This experiment was set up identically to the first MCIM experiment from Section 8.2, except that the TCIM compact representation scheme was used in place of the MCIM representation. The results are shown in Fig. 15.

Fig. 15a shows that as the approximation threshold increases, the total number of intensity matrices required to specify the model decreases, and the variance increases. Furthermore, Fig. 15b shows that the error observed by running inference on a compacted model increases with the size of the approximation threshold. In other words, the model may be represented more compactly by accepting worse approximations.

Note that the savings in space requirements shown in Fig. 15a are less dramatic than the corresponding MCIM results shown in Fig. 9a. This is due to the difference in representational capabilities. A single discrete function is capable of fully encoding all clusterings, while a tree structure cannot necessarily capture all of the same information. Despite this, a tree structure is a more efficient means of encoding information when compared to a single discrete mapping function. See Fig. 15c, which shows that inference over the compacted representation is significantly faster than the original network due to the efficiency of the TCIM data structure.

10.1.2. Network structure

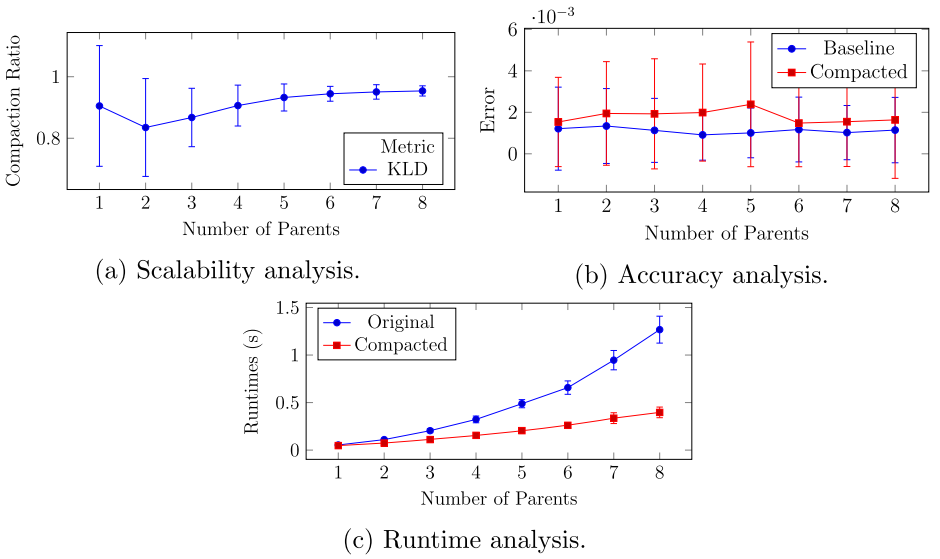
The next experiment was designed to investigate the impact of network size when using a TCIM representation. This experiment was set up to match the second MCIM experiment from Section 8.2, except that rather than using an MCIM representation, a TCIM was used to model the child node in the network. The results are shown in Fig. 16.

As shown by Fig. 16a, the compaction ratio initially decreases, and then steadily increases as the size of the networks increase. This is likely because the TCIM representation relies on pruning portions of the tree, and the unstructured nature of the data makes it unlikely that large branches will be pruned when networks are large. The reduction in variance indicates that the tree structure is consistently providing a small amount of compaction for large networks. In terms of the error introduced by the compact representation, Fig. 16b shows no statistical significance between the two models, although just as shown by the MCIM experiments, there may be a small error introduced by the constant approximation threshold.

Despite the minimal improvements to space requirements, the efficient tree structure provides substantial improvements in terms of runtimes. Fig. 16c shows the runtimes in seconds for inference over a TCIM compacted model as well as a baseline model with no compaction. While the runtimes for the uncompacted models increase exponentially with the network size, the compacted model times appear almost linear. In addition, the variance for inference over the original model increases as shown by the error bounds, while the variance for the compacted model remains essentially negligible.

10.2. Structured data experiments

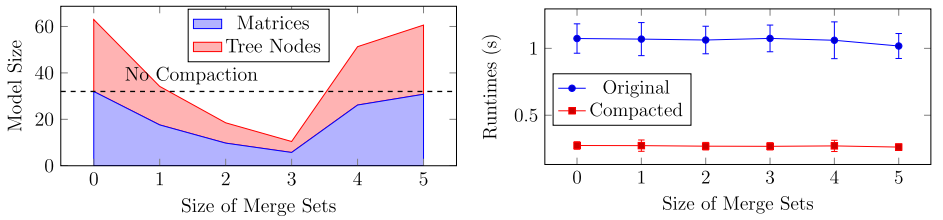
The structured data experiments in this section used models designed as described previously in Section 8.2. Specifically, random models were generated, which were then modified to contain approximate equivalence using the notion of merge sets. The number of merge sets and the size of the merge sets varied based on experiments.



(a) Scalability analysis. (b) Accuracy analysis.

(c) Runtime analysis.

Fig. 16. Impact of network structure with unstructured data using a TCIM representation.



(a) Space analysis. (b) Time analysis.

Fig. 17. Impact of the size of merge sets with structured data using a TCIM representation.

10.2.1. Merge set size

The results of the merge set size experiment are shown in Fig. 17. In this case, Fig. 17a shows the size of the model in terms of both the number of intensity matrices (the lower blue region), and the number of nodes in the tree that stores the intensity matrices (the upper red region). Another interpretation is that the lower region shows the number of leaf nodes in the tree, while the upper region shows the number of interior nodes. With a set size of zero, the number of leaf nodes match the baseline model, which intuitively makes sense given that there is no compaction occurring. Just as in the MCIM experiments, a set size of three produces the most compact models.

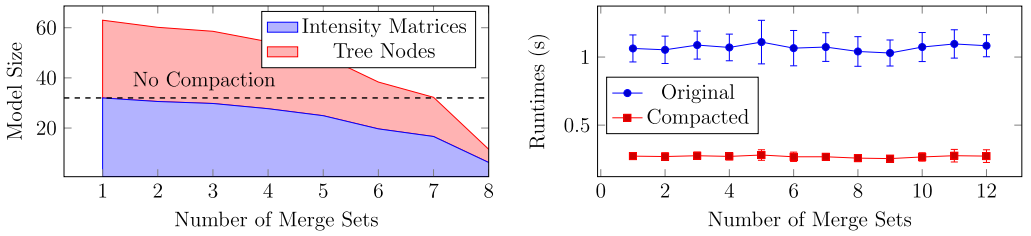
Fig. 17b shows the runtimes for inference over the baseline and compacted models. Not only is inference over the TCIM compacted model substantially faster, the variance is reduced by switching to the compact representation as well, as indicated by the error bars. This improvement in inference speed does not appear to be affected by the merge set size, indicating that the efficiency of the TCIM representation overpowers any effect that model size has on inference runtimes.

10.2.2. Number of merge sets

Just as with the MCIM experiments, the number of merge sets was varied from one to eight to account for all possible state combinations of the three parents. Fig. 18a shows the size of the model in terms of the number of intensity matrices and tree nodes. As the number of merge sets increase, the TCIM structure is able to encode increasingly more compact models. This follows the same trend as in the MCIM experiment, showing that in either case, the compact representations are more efficient when more states can be merged. Fig. 18b shows runtimes for inference run over the original and compacted models. Once again, the TCIM compacted model results in substantially reduced runtimes with less variation. This indicates that the TCIM representation improves inference speeds, regardless of the type of structure exhibited in the data.

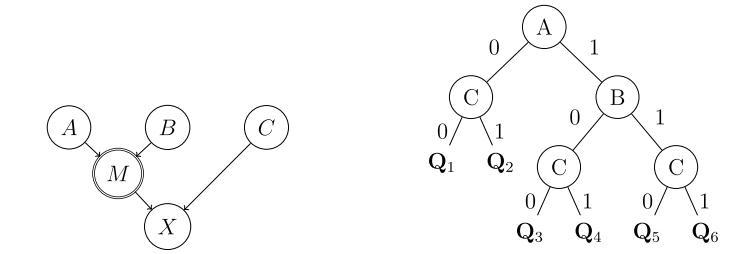
11. Comparison between MCIMs and TCIMs

There are many different potential cluster possibilities when grouping intensity matrices. This section compares the representational capabilities of MCIMs and TCIMs by inspecting different types of clusterings. In this section, a non-trivial



(a) Space analysis. (b) Time analysis.

Fig. 18. Impact of the number of merge sets with structured data using a TCIM representation.



(a) MCIM for six clusters. (b) TCIM for six clusters.

Fig. 19. Compact structures for CIM clustering.

MCIM refers to a composition of functions containing a non-empty context set. A single mapping function with inputs $\mathbf{V}$ equal to the entire parent set $\mathbf{U}$ is capable of fully representing any possible clustering, although as discussed in Section 7, this type of representation is itself too complex to provide any computational benefits. As such, only non-trivial mappings are considered in the following.

Theorem 1. *There are clusterings that can be represented fully and equivalently using either a non-trivial MCIM or a TCIM.*

Proof. To prove this claim, it is sufficient to provide a single example where this statement holds. Consider a network containing a node X and three parents $\mathbf{U} = \{A, B, C\}$, each with two states. A potential clustering of the CIM for X is as follows.

C_1:	$\mathbf{Q}_{X\|\{a_0,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_0\}}$
C_2:	$\mathbf{Q}_{X\|\{a_0,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_1\}}$
C_3:	$\mathbf{Q}_{X\|\{a_1,b_0,c_0\}}$	
C_4:	$\mathbf{Q}_{X\|\{a_1,b_0,c_1\}}$	
C_5:	$\mathbf{Q}_{X\|\{a_1,b_1,c_0\}}$	
C_6:	$\mathbf{Q}_{X\|\{a_1,b_1,c_1\}}$	

The original CIM consists of $2^3 = 8$ intensity matrices, which are then condensed down to six clusters. Fig. 19a shows a MCIM that encodes this clustering, while Fig. 19b shows a TCIM that captures the same clustering. Here, the mapping variable M in the MCIM reduces the four state combinations of $\{A, B\}$ down to three, meaning that node X contains $3 \times 2 = 6$ intensity matrices, one for each of the clusters. Similarly, the tree structure has six leaf nodes corresponding to each cluster, indicating that these representations were in fact able to encode the entire clustering. □

Theorem 2. *There are clusterings that can be represented fully using a non-trivial MCIM but not a TCIM.*

Proof. Again, a single example is sufficient to prove that there are clusterings which can be represented using MCIMs, but not TCIMs. Consider the clustering presented in the previous section, but now assume that B has three states rather than two, resulting in four new matrices that do not belong to any of the previous clusters. This new clustering is as follows:

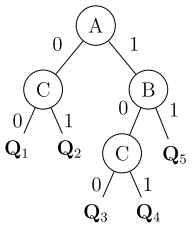


Fig. 20. Tree-structured CIM.

C_1:	$\mathbf{Q}_{X\|\{a_0,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_0\}}$
C_2:	$\mathbf{Q}_{X\|\{a_0,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_1\}}$
C_3:	$\mathbf{Q}_{X\|\{a_1,b_0,c_0\}}$	
C_4:	$\mathbf{Q}_{X\|\{a_1,b_0,c_1\}}$	
C_5:	$\mathbf{Q}_{X\|\{a_1,b_1,c_0\}}$	
C_6:	$\mathbf{Q}_{X\|\{a_1,b_1,c_1\}}$	
C_7:	$\mathbf{Q}_{X\|\{a_0,b_2,c_0\}}$	
C_8:	$\mathbf{Q}_{X\|\{a_0,b_2,c_1\}}$	
C_9:	$\mathbf{Q}_{X\|\{a_1,b_2,c_0\}}$	
C_{10}:	$\mathbf{Q}_{X\|\{a_1,b_2,c_1\}}$	

These clusters cannot be represented by any tree structure. This is because there are three states of B, and these states are not all equal in any context of A and C. The MCIM framework does not require that all states be equal and provides a method for consolidating a subset of states. In this example, the tree structure from Fig. 19b is sufficient to describe the clustering described in this section. Here, M maps state combinations $\{a_0, b_0\}$ and $\{a_0, b_1\}$ to a single state and maps the remaining four state combinations to their own states. This results in a five state mapping variable in combination with the unmodified two state variable C, resulting in a total of 10 possible state combinations for the parents of variable X. This matches the clustering described in this section where two of the original twelve intensity matrices in the network have been consolidated using the MCIM framework. A TCIM representation would require a full tree with no pruning, necessitating the storage of all twelve of the original intensity matrices. □

This example suggests that the MCIM framework is better suited for representing CIMs when parent variables have a large number of states. As the number of states grows, it becomes increasingly less likely that all intensity matrices will be equal across all states for a given context. This means that TCIMs are likely to provide little or no benefit in these cases. An MCIM is capable of capturing equivalencies over a subset of the total number of states, which is a more reasonable requirement when there is a large number of states.

Theorem 3. *There are clusterings that can be represented fully using a TCIM but not a non-trivial MCIM.*

Proof. In this case, the example starts with the same clustering from Section 1, except that clusters C_5 and C_6 are now close enough that they can be merged. The resulting cluster set is as follows, where $C_5 \approx C_6$.

C_1:	$\mathbf{Q}_{X\|\{a_0,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_0\}}$
C_2:	$\mathbf{Q}_{X\|\{a_0,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_1\}}$
C_3:	$\mathbf{Q}_{X\|\{a_1,b_0,c_0\}}$	
C_4:	$\mathbf{Q}_{X\|\{a_1,b_0,c_1\}}$	
C_5:	$\mathbf{Q}_{X\|\{a_1,b_1,c_0\}}$	$\mathbf{Q}_{X\|\{a_1,b_1,c_1\}}$

This clustering cannot be represented efficiently and entirely using the MCIM framework. The consolidation of the matrices in clusters C_1 and C_2 can be achieved as described in the previous section, but this does not hold for the matrices in C_5. The issue is that the equivalence observed in C_5 does not hold across all contexts; therefore, it cannot be captured by mapping variables except in the trivial case where a variable is introduced as a child for all parent A, B and C. Conversely, a TCIM can describe this clustering compactly without issue (Fig. 20) where the five leaf nodes correspond to the five clusters, thereby providing a minimal representation. □

This example suggests that a TCIM may work better for representing clusterings when there are a large number of parent variables. This is because large parent sets make it increasingly less likely that multiple instantiations will be equivalent across *all* contexts. As the number of parents grows, the size of the subject and context sets grows exponentially, likely making MCIMs a poor choice for networks with large parent sets. This shortcoming is not shared by tree structures, that are likely able to exploit context-based independence regardless of the number of parent nodes.

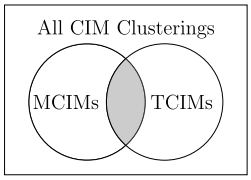


Fig. 21. Venn Diagram depicting CIM clusterings that can be represented using MCIMs and TCIMs.

Theorem 4. *There are clusterings that cannot be represented fully using either a non-trivial MCIM or a TCIM.*

Proof. The previous two cases may be combined to form an example of a clustering that cannot be fully represented by either MCIMs or TCIMs. Specifically, a clustering can be produced that contains a three state variable B, as well as a clustering of $\mathbf{Q}_{X|\{a_1,b_1,c_0\}}$ and $\mathbf{Q}_{X|\{a_1,b_1,c_1\}}$. The combined clustering is shown below.

C_1:	$\mathbf{Q}_{X\|\{a_0,b_0,c_0\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_0\}}$
C_2:	$\mathbf{Q}_{X\|\{a_0,b_0,c_1\}}$	$\mathbf{Q}_{X\|\{a_0,b_1,c_1\}}$
C_3:	$\mathbf{Q}_{X\|\{a_1,b_0,c_0\}}$	
C_4:	$\mathbf{Q}_{X\|\{a_1,b_0,c_1\}}$	
C_5:	$\mathbf{Q}_{X\|\{a_1,b_1,c_0\}}$	$\mathbf{Q}_{X\|\{a_1,b_1,c_1\}}$
C_6:	$\mathbf{Q}_{X\|\{a_0,b_2,c_0\}}$	
C_7:	$\mathbf{Q}_{X\|\{a_0,b_2,c_1\}}$	
C_8:	$\mathbf{Q}_{X\|\{a_1,b_2,c_0\}}$	
C_9:	$\mathbf{Q}_{X\|\{a_1,b_2,c_1\}}$	

The three state issue from Section 2 prevents complete representation of the clustering using a TCIM. Similarly, the combined matrices in cluster C_5 have the same issue seen in Section 3, in that the equivalence does not hold across all contexts. For this reason, an MCIM is unable to encode the clustering efficiently unless all parent variables are included as inputs to the discrete mapping function. An MCIM representation of the clustering requires storing 10 intensity matrices, and a TCIM will need to store 11 matrices at the leaves. Although the MCIM appears to be a slightly better choice, neither representation is able to represent the minimal nine intensity matrices defined by the clusters. This demonstrates that clusterings exist that can be captured only partially using MCIMs and TCIMs. □

A graphical depiction of the results is shown in Fig. 21. The shaded area between the circles indicates clusterings that can be captured entirely by either non-trivial MCIMs or TCIMs, shown by Theorem 1. The left circle represents clusterings that can be represented using non-trivial MCIMs (Theorem 2), while the right circle represents clusterings that can be represented using TCIMs (Theorem 3). Finally, the surrounding rectangle represents the entire space of clusterings, and the area outside of the circles is the set of clusterings that cannot be represented with either non-trivial MCIMs or TCIMs, as discussed in the proof for Theorem 4.

12. Conclusion

In this work, we introduced two novel methods for imposing additional structure on top of the already factored CTBN representation. Specifically, this was achieved by defining the MCIM and the TCIM: two distinct approaches to representing conditional CTMPs compactly. The MCIM exploits the notion of context-independent equivalence, allowing the conditional CTMP to be factored using a composition of functions. Similarly, the TCIM takes advantage of continuous-time context-specific independence and uses a tree structure to factor a conditional CTMP. In both cases, the compact structures make use of intensity matrix clustering as a preprocessing step, which allows for approximate compact representations as well as exact. Experiments were conducted that evaluated the capabilities of both the MCIM and TCIM, and it was proved that the classes of conditional CTMPs that can be represented by MCIMs and TCIMs intersect, but are not equivalent. These two compact representations provide a means of combating scalability concerns, especially when data exhibits strong context-independent equivalence or continuous-time context-specific independence.

For future work, we would like to look into hybrid compact structures. As proven, there are conditional CTMPs that cannot be factored fully using either the MCIM or TCIM representation. By combining these approaches, and possibly other compact structures designed for Markov processes, it may be possible to achieve an even more effective representation. We also hope to identify improved techniques for choosing a parameter τ when clustering, or learning the decision tree structure that maximize space efficiency while minimizing model differences according to user-defined cost metrics. Finally, we hope to adapt existing inference algorithms to work on the compact representations directly, potentially allowing for even more efficient inference.

Conflict of interest statement

There is no conflict of interest statement.

Acknowledgements

We would like to thank members of the Numerical Intelligent Systems Laboratory at Montana State University.

References

[1] W. Bian, D. Tao, Harmonic mean for subspace selection, in: 19th International Conference on Pattern Recognition (ICPR), IEEE, 2008, pp. 1–4.
[2] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence (UAI), Morgan Kaufmann Publishers Inc., 1996, pp. 115–123.
[3] P. Buchholz, G. Ciardo, S. Donatelli, P. Kemper, Complexity of Kronecker Operations on Sparse Matrices with Applications to the Solution of Markov Models, Tech. rep., Institute for Computer Applications in Science and Engineering, Hampton, VA, United States, 1997.
[4] A. Cutler, O.I. Cordero-Braña, Minimum Hellinger distance estimation for finite mixture models, J. Am. Stat. Assoc. 91 (436) (1996) 1716–1723.
[5] T. Dean, K. Kanazawa, A model for reasoning about persistence and causation, Comput. Intell. 5 (2) (1989) 142–150.
[6] Y. Fan, C.R. Shelton, Sampling for approximate inference in continuous time Bayesian networks, in: International Symposium on Artificial Intelligence and Mathematics, 2008.
[7] Y. Fan, J. Xu, C.R. Shelton, Importance sampling for continuous time Bayesian networks, J. Mach. Learn. Res. 11 (Aug. 2010) 2115–2140.
[8] M. Fujita, P.C. McGeer, J.-Y. Yang, Multi-terminal binary decision diagrams: an efficient data structure for matrix representation, Form. Methods Syst. Des. 10 (2–3) (1997) 149–169.
[9] A.L. Gibbs, F.E. Su, On choosing and bounding probability metrics, Int. Stat. Rev. 70 (3) (2002) 419–435.
[10] M. Halkidi, Y. Batistakis, M. Vazirgiannis, Cluster validity methods: Part I, SIGMOD Rec. 31 (2) (2002) 40–45.
[11] M. Halkidi, Y. Batistakis, M. Vazirgiannis, Clustering validity checking methods: Part II, SIGMOD Rec. 31 (3) (2002) 19–27.
[12] A.N. Kolmogorov, Foundations of Probability, Chelsea Publishing Company, New York, 1933.
[13] P.W. Laud, J.G. Ibrahim, Predictive model selection, J. R. Stat. Soc. B 57 (1) (1995) 247–262.
[14] P. Máša, T. Kocka, Finding Optimal Decision Trees, Ph.D. thesis, Stanford University, 2007.
[15] A. Mesaros, T. Virtanen, A. Klapuri, Singer identification in polyphonic music using vocal separation and pattern recognition methods, in: The International Society of Music Information Retrieval, 2007, pp. 375–378.
[16] F. Murtagh, A survey of recent advances in hierarchical clustering algorithms, Comput. J. 26 (4) (1983) 354–359.
[17] U. Nodelman, C. Shelton, D. Koller, Continuous time Bayesian networks, in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI), 2002, pp. 378–387.
[18] C.R. Shelton, G. Ciardo, Tutorial on structured continuous-time Markov processes, J. Artif. Intell. Res. 51 (2014) 725–778.
[19] R.R. Sokal, F.J. Rohlf, The comparison of dendrograms by objective methods, Taxon (1962) 33–40.
[20] A. Srinivasan, T. Ham, S. Malik, R.K. Brayton, Algorithms for discrete function manipulation, in: IEEE International Conference on Computer-Aided Design (ICCAD-90), 1990, pp. 92–95.
[21] L. Sturlaugson, J.W. Sheppard, Factored performance functions with structural representation in continuous time Bayesian networks, in: Florida Artificial Intelligence Research Society (FLAIRS) Conference, 2014, pp. 512–517.
[22] L. Sturlaugson, J.W. Sheppard, The long-run behavior of continuous time Bayesian networks, in: Conference on Uncertainty in Artificial Intelligence (UAI), 2015, pp. 842–851.
[23] L. Sturlaugson, J.W. Sheppard, Sensitivity analysis of continuous time Bayesian network reliability models, Int. J. Uncertain. Quantificat. 3 (1) (2015) 346–369.
[24] G.J. Szekely, M.L. Rizzo, Hierarchical clustering via joint between-within distances: extending Ward's minimum variance method, J. Classif. 22 (2) (2005) 151–183.
[25] J. Weiss, S. Natarajan, D. Page, Multiplicative forests for continuous-time processes, in: Advances in Neural Information Processing Systems, 2012, pp. 458–466.
[26] W.A. Woodward, P. Whitney, P.W. Eslinger, Minimum Hellinger distance estimation of mixture proportions, J. Stat. Plan. Inference 48 (3) (1995) 303–319.
[27] V.M. Zolotarev, Probability metrics, Theory Probab. Appl. 28 (2) (1983) 264–287.