Data Mining and Knowledge Discovery (2022) 36:240–294
https://doi.org/10.1007/s10618-021-00807-y

Temporal state change Bayesian networks for modeling of evolving multivariate state sequences: model, structure discovery and parameter estimation

Artur Mrowca[1] · Florian Gyrock[2] · Stephan Günnemann[2]

Received: 31 August 2019 / Accepted: 11 October 2021 / Published online: 1 November 2021

Abstract

Many systems can be expressed as multivariate state sequences (MSS) in terms of entities and their states with evolving dependencies over time. In order to interpret the temporal dynamics in such data, it is essential to capture relationships between entities and their changes in state and dependence over time under uncertainty. Existing probabilistic models do not explicitly model the evolution of causality between dependent state sequences and mostly result in complex structures when representing complete causal dependencies between random variables. To solve this, Temporal State Change Bayesian Networks (TSCBN) are introduced to effectively model interval relations of MSSs under evolving uncertainty. Our model outperforms competing approaches in terms of parameter complexity and expressiveness. Further, an efficient structure discovery method for TSCBNs is presented, that improves classical approaches by exploiting temporal knowledge and multiple parameter estimation approaches for TSCBNs are introduced. Those are expectation maximization, variational inference and a sampling based maximum likelihood estimation that allow to learn parameters from partially observed MSSs. Lastly, we demonstrate how TSCBNs allow to interpret and infer patterns of captured sequences for specification mining in automotive.

Responsible editor: Michelangelo Ceci.

This work was supported by the BMW Group.

✉ Artur Mrowca
artur.mrowca@bmw.de

Florian Gyrock
florian.gyrock@tum.de

Stephan Günnemann
guennemann@in.tum.de

1 Bayerische Motoren Werke AG, Knorrstr. 147, 80788 München, Germany

2 Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany

Keywords Temporal Bayesian networks · Multivariate state sequences · Probabilistic processes · Latent variable models

1 Introduction

Many domains, including medicine (Orphanou et al. 2016) and automotive (Mrowca et al. 2018b) require to capture temporal system behavior for diagnosis and prediction tasks. Such systems are increasing in complexity, as functional variety and inter connectivity of components grow. Gaining insights of its underlying processes is essential to allow for better understanding, functional optimization and fault reduction.

To achieve this in a data-driven manner, a common approach is to capture procedural behavior in models, that are trained from observations of recorded system executions. This type of behavior can be represented along multiple dimensions (which we call Temporal Variables (TV)) as processes that change its state over time. For example, the behavior to inspect might be the activation process of a sprinkler system, which consists of the three TVs *grass*, *sprinkler* and *rain*, as it is illustrated in Fig. 1. Each execution of this process forms an observed Multivariate State Sequence (MSS), e.g. Fig. 1 shows the MSS $\langle g = dry, s = no, r = no, s = yes, g = wet, r = yes, s = no \rangle$. Other runs of this process might have produced different outcomes. By using multiple such observed MSSs, a model is trained to represent this process.

MSSs of behavioral processes contain a high number of execution variants in complex systems and are usually noisy, incomplete and imprecise in real life. Therefore, models that represent such multidimensional processes need to be robust, to compactly represent the process and to allow for inference in multiple dimensions under uncertainty.

Existing models do have major limitations in the representation of such processes, which is why we introduce a novel model in this work.

First, related models are presented in Sect. 1.1. An overview of our novel model is found in Sect. 1.2 and an outline of this work in Sect. 1.3.

1.1 Related work: existing models

Process models First, existing models are the ones used in the field of Process Mining (e.g. Petri Nets). Those representations assume one dimensional sequences. This yields massive model sizes that are hard to interpret in the given scenario, where multiple execution variants and dimensions are possible. This is amplified by the fact that, each occurrence and state that a TV is in, does produce an additional node in the resulting model. Assuming that any variant can occur in any state, in the example of Fig. 1, this would give $3 \cdot 2 + 3 \cdot 2 + 2 \cdot 2 = 16$ nodes that all could be interlinked. Further, those methods do mostly use frequency based filtering, rather than statistical methods, for complexity reduction during model discovery. This yields a less precise structure. Inference is still possible with those models, while it is not practical as high complexity needs to be dealt with, no uncertainty is captured and dimensional information is excluded.

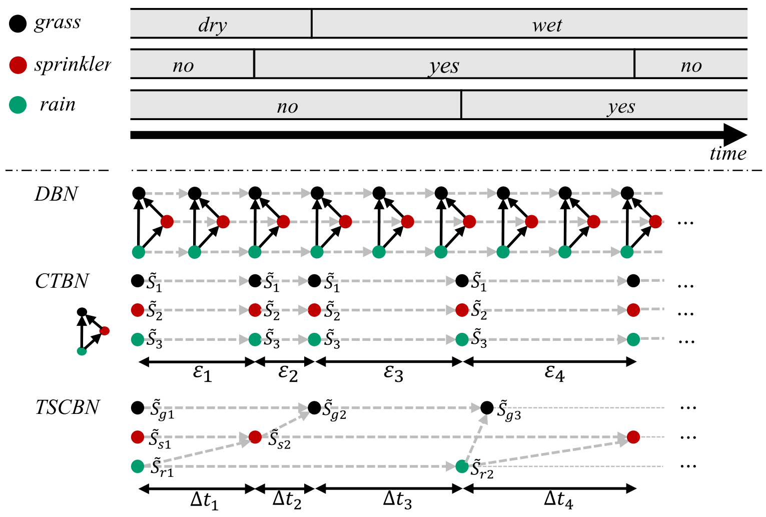


Fig. 1 A MSS with 3 temporal variables for the process of wetting grass, and models to generate it, are shown. If the grass is *dry* the sprinkler turns *on*, if it is not *raining*. Once it starts *raining* the sprinkler turns *off*. The same model could produce a sequence where the rain is falling throughout the process. Then, the sprinkler would never have turned *on*. Temporal State Change Bayesian Networks provide a compact yet expressive representation for such scenarios. $\tilde{S}_{g3}$ indicates the latent state change $\langle wet, wet \rangle$, which is not observed (i.e. *latent*) here, while in another constellation an observed change, e.g. $\langle wet, dry \rangle$, might have occurred

Temporal Bayesian networks (BN) A more promising direction for modeling MSS processes are temporal BNs, as those allow for more compact representations, multi-dimensionality and uncertainty. This is why we also chose to design our novel model on the basis of BNs.

Time-sliced BNs: The first group of models includes time-sliced BNs such as discrete time nets (Dean et al. 1989) with static structures and its dynamically adjusting extensions (Olesen et al. 2006) and Dynamic Bayesian Networks (DBN), that break the Markovian assumption (Tucker 2001) or the stationarity assumption (Song et al. 2009). The application of DBNs to the given scenario is shown in Fig. 1. It can be seen that a static dependency structure is used, that is repeated in regular time intervals. Main caveats of such networks include its static structure, and its high overhead due to repetition of structures.

Interval-based BNs: The group of interval-based BNs models temporal events or intervals as states of nodes. A subgroup of those BNs defines the type of event as state, and the interval in which the event occurred as outcome, in a node. There nodes represent irreversible events whose outcomes are a cross product of discretized intervals and state outcomes of events (Arroyo-Figueroa et al. 1999; Galán et al. 2000). Problems of such models include, that for higher precision this leads to huge outcome spaces and events are irreversible. Other models represent a certain event only, with times of occurrence as outcome (Galán et al. 2002).

Temporal Bayesian Network of Events (TBNE) that were introduced in Arroyo-Figueroa et al. (2005) are comparable to Temporal State Change Bayesian Networks

(TSCBN). Similar to TSCBNs, there, intervals are modeled as tuples of time of occurrence of a state change and states of the next interval including the option of remaining in the same state. However, TBNEs omit the following aspects, which are solved in TSCBNs. First, interval lengths are discretized which allows for less temporal precision. Second, no notion for representation of dynamic causal relations between states of different TVs is defined. Third, the model does not allow to model causal dependence of state transitions that remain in the same state, which is solved in TSCBNs using latent sequences. Networks, such as Modifiable Temporal Belief Networks (MTBN) (Aliferis et al. 1996) allow to model intervals in terms of nodes for *start*, *end* and *duration* of an interval. There again each node represents the time and duration of one event only. Kwon and Suh (2012) the modeling of temporal information is accomplished by using Hybrid Bayesian Networks (HBN) consisting of discrete nodes for the values and continuous nodes for the time delays. However, rather than modeling dynamics of a process as TSCBNs do, those model dependence between time and state statically.

Process based models To model continuous time under uncertainty Markov Processes, Continuous-Time Markov Processes or other processes are used. Nodelman et al. (2002) Nodelmann introduced Continuous Time Bayesian Networks (CTBN), whose application on the given scenario is also shown in Fig. 1. This model uses Markov Processes that depend on the structure of one static BN to determine consequent states and time intervals in dependence of states a TV is in in time. Those CTBNs have been extended in terms of more general transition times, explicit negative evidence (Gopalratnam et al. 2005) or by adding static chance nodes of BNs (Portinale and Codetta-Raiteri 2009). To model event streams, in Bhattacharjya et al. (2020) Event-Driven Continuous Time Bayesian Networks, where proposed that model event occurrences modeled as a multivariate point process and states as Markov processes. In further works, event streams are modeled with Poisson cascades (Simma and Jordan 2012) or graphical event models (Gunawardana and Meek 2016).

As processes are used in such models, the temporal behavior is modeled implicitly. Thus, no explicit inference can be performed on this model and especially for long processes processes get impossible to model as the same matrices and structures are used to model all stages of the process. Tawfik et al. (2000) Tawfik considered events as distributions over time. This approach unlike us does not model defined processes of intervals, but rather static causal dependencies of continuously evolving processes.

General probabilistic networks Apart from BNs in Causal Probabilistic Networks (Berzuini 1990) event occurrence and its times of occurrences are modeled separately. That means, dynamic causal influence of states on duration of intervals of TVs is not included (Ryabov et al. 2004). Probabilistic Temporal Interval Networks nodes are temporal intervals and edges are uncertain interval relations modeled in terms of Allen's relations. Thus, intervals are considered rather qualitatively (e.g. as *duration*, *before*) than quantitatively.

Specialized models Further, in many domains specialized models were proposed. In reliability engineering Object Oriented BNs (De Carlo et al. 2013), Markov Chains and Event Sequence Diagrams (Swaminathan et al. 1999) were applied for temporal modeling under uncertainty. Those approaches focus on object interactions and fault detection without general extensibility and applicability to interval modeling. Models of temporally evolving dependencies further include the approach proposed in Hallac et al. (2017), where evolving Markov Random Field based models are used for time-series clustering or in Gu et al. (2017) and Corneli et al. (2015), where temporal evolution in social networks is modeled. However, the further is optimized for time-series, while the latter is focused on graph data rather than MSSs.

1.2 Overview of TSCBNs

Those approaches do have multiple limitations when modeling MSSs, which are overcome by the model proposed in this work. As a basis to discuss this, the idea of TSCBNs is given first, before, next, limitations of existing models are pointed out.

TSCBNs As Figure 1 shows, TSCBNs are Hybrid BNs, that model each state change of a MSS as a node (preliminaries on Bayesian Networks can be found in Appendix A). Each of those nodes, consists of a discrete state node and a continuous time node. The further, represents the states a MSS changes to, given any previous MSS state that it is causally dependent on, e.g. $\tilde{S}_{g2}$ is causally influenced by $\tilde{S}_{s2}$. If $\tilde{S}_{s2}$ was given a state change $\langle no, yes\rangle$ the likelihood of $\tilde{S}_{g2}$ to be in the state change $\langle dry, wet\rangle$ is higher than in the case where $\tilde{S}_{s2}$ was given the state change $\langle no, no\rangle$.
Continuous time nodes represent the temporal distribution which determines when the state change occurs, depending on state nodes that causally influence this state change (i.e. that have an edge to this node).

Thus, TSCBNs model a process as a whole, with edges at times where causality is given in the process and nodes at each position where a state change might occur with a sufficiently high likelihood.

Limitations of exiting approaches First, other approaches are not optimally compact and require a high number of parameters. Process models require complex structures, time-sliced BNs require redundant repetition of structure, interval-based approaches are of qualitative nature (e.g. MTBNs Aliferis et al. 1996) or explode with process lengths (Galán et al. 2002). TSCBNs and TBNEs (Arroyo-Figueroa et al. 2005) are similarly compact, as only likely changes are modeled. Markov-based approaches do not model the procedure explicitly, while being compact as well.

Second, good inference requires an unrolled model, as this allows to directly read of behavioral patterns and to more precisely express multidimensional processes. Apart, from Process models, TBNEs and DBNs are the only comparable unrolled representations. TSCBNs are also unrolled.

Third, existing models do not allow for multidimensional procedural inference. To assess uncertainty of a subpart of a process, given what happened in the first part of this process, it is essential to be able to provide evidence. DBNs allow for this similar to TSCBNs.

Fourth, causal dependencies are likely to change throughout a modeled process. Therefore, causality needs to be represented between state changes that actually influence one another. Existing models redundantly store a static structure (e.g. DBNs, CTBNs) or are of high complexity (e.g. process models). TSCBNs are compact while at the same time representing causality only at steps in a process where it is existent.

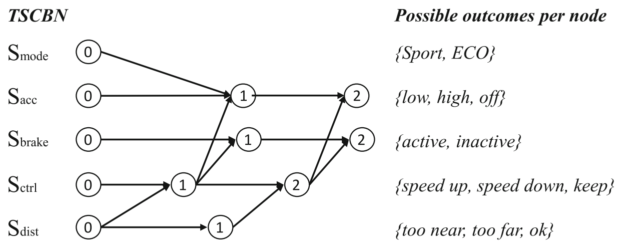


Fig. 2 Here, the compact TSBCN representation of the adaption process of the active cruise control system of a car is shown. It consists of 5 TVs. The distance, the controller, the brakes, the acceleration and the driving mode. If e.g. the distance is too high (i.e. node 0 of S_{dist} has outcome *too far*), the controller will send an acceleration command (i.e. node 1 of S_{ctrl} is *speed up*), which activates the high acceleration (i.e. engine, node 1 of S_{acc} is *high*) in Sport mode (i.e. node 0 of S_{mode} is *Sport*) until the distance is in an acceptable range (i.e. node 1 of S_{dist} is *ok*) in which case it will deactivate (i.e. node 2 of S_{ctrl} is *keep*) the acceleration (i.e. node 2 of S_{acc} is *off*)

Lastly, per TV only *state changes* are observed. That is, for the case where an interval remains in the same state, the transition between those intervals is not observed, while it still might have a causal influence on other states. In the example of Fig. 1 the TV grass has the true (but unobserved) state sequence $\langle dry, wet, wet\rangle$, as once the rain starts falling it has a causal influence on the TV grass' state by making it *wet*. However, what is actually observed in this case are only the state changes $\langle dry, wet\rangle$. Thus, the actual sequence $\langle dry, wet, wet\rangle$ is *latent*. Estimating this latent sequence from observations is challenging as multiple valid sequences could have been produced (e.g. $\langle dry, dry, wet\rangle$). TSCBNs are the first model to solve this problem.

All of those limitations and how those are overcome by TSCBNs are discussed in more detail in Sect. 2.5.

1.3 Outline

This work provides a novel approach to represent MSSs as TSCBNs, as well as methods to learn both the structure and parameters of TSCBNs from MSSs.

In Sect. 2 a definition of the model TSCBNs is given. Learning of structure is presented in Sect. 3 and learning of parameters discussed in Sect. 4. Lastly, in Sect. 5 TSCBNs and its learning methods are evaluated. In Sect. 6 the usage of TSCBNs is discussed in a real world scenario.

Running example To improve comprehensibility we use a simplified example process in the following sections. This is the adaption process of the automatic cruise control, that, while driving, ensures that the distance to the preceding car is kept stable. Figure 2 shows the TSCBN that represents this adaption process.

Definition of multivariate state sequences As illustrated in Fig. 3 for our running example, a MSS consists of multiple TVs. Each TV has a state it is in at any point in time, e.g. the TV S_{dist} indicates the state of the distance sensor at any time. In Fig. 3 the distance has an initial state *too near*, which activates the controller to adjust the speed. The controller in turn activates the brakes. Any state change is causally dependent on its previous state. Further, if states change and which states TVs change

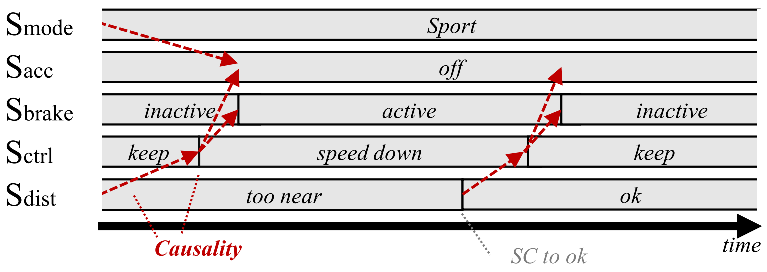


Fig. 3 A possible MSS of the running example of Fig. 2 is shown. It shows one possible outcome for the adaption process of the active cruise control, i.e. the process of braking until an acceptable distance is given, if a preceding car is too near. Notably, if the controller had been in state *speed up*, the acceleration would have changed to *high*, i.e. here, causality between S_{ctrl} and S_{acc} is given at this stage

to might depend on other TVs, e.g. in Fig. 3 the brake state might more likely change from *inactive* to *active* if the state of the controller is *speed down*.
Such MSSs were introduced in Batal and Sacchi (2009). Here, MSSs M are interdependent sequences $E_i = (s, e, S_k, s_j)$ of intervals of TVs S_k, where all TVs are in a defined state at any time, with start times s, end times e and states s_j they are in. Also, each TV is in a state at all times and only state changes are observed. Formally that means that

$$M = \langle E_1, ..., E_l \rangle : E_i.s \leq E_{i+1}.s, E_i.e = E_{i+1}.s$$

2 Model

In this section we introduce TSCBNs. For this, first, TSCBNs are defined formally in Sect. 2.1 based on the example in Figs. 3 and 4. A comprehensive explanation of those models is presented in Sects. 2.2 and 2.3. Second, a comparison to existing approaches is given in Sect. 2.5.

2.1 Formal definition

Let's assume a set $\mathcal{S}$ of TVs S_i that each temporally evolves over a set of states $\Xi_i = \{s_{i1}, s_{i2}, ...\}$, which can be dynamically interdependent, e.g. in Fig. 3 there are 5 TVs, where TV S_{brake} has $\Xi_{brake} = \{inactive, active\}$.

Every *TSCBN* is a HBN $B = (G, \Theta)$ with parameter set Θ, DAG $G = (N, E)$, nodes N and directed edges E. While the inverse does not hold, a HBN is a TSCBN under the following conditions.

– Exactly one discrete node $v_{ik} \in V$ (called *state node*) and one continuous node $\Delta t_{ik} \in T$ (called *temporal node*) represent the possible state change $* \mapsto s_{ij}$ of a TV S_i from its previous state $*$ to its next state s_{ij}, where $N = V \cup T$.

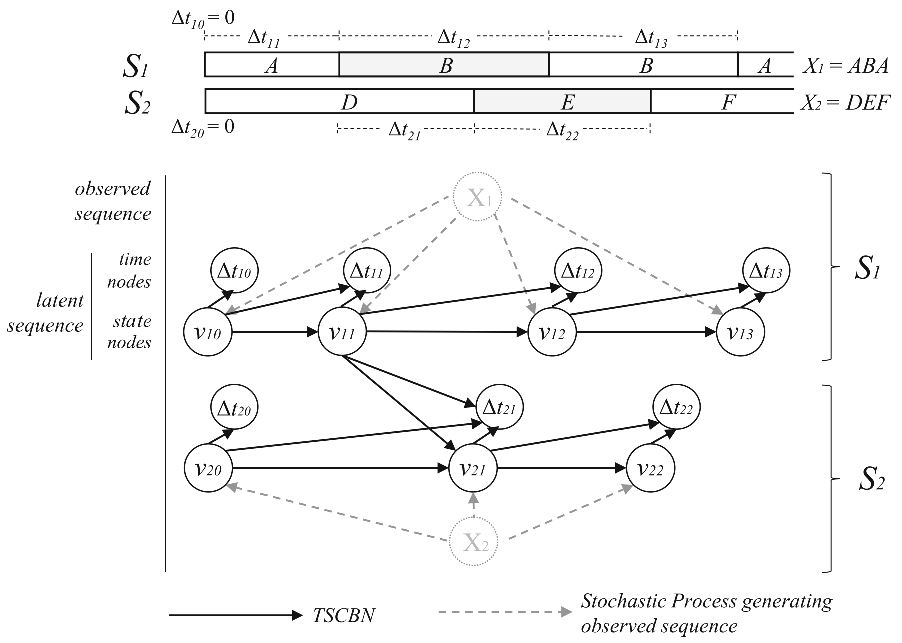


Fig. 4 Two TVs S_1 and S_2 are illustrated. The top part shows a MSS that is *generated* by the TSCBN shown in the lower part. X_1 and X_2 indicate the *observed sequence* of state changes, which are generated by the true *latent sequences* $\langle(v_{10}, \Delta t_{10}), (v_{11}, \Delta t_{11}), (v_{12}, \Delta t_{12}), (v_{13}, \Delta t_{13})\rangle$ and $\langle (v_{20}, \Delta t_{20}), (v_{21}, \Delta t_{21}), (v_{22}, \Delta t_{22})\rangle$. Note that a temporal-causal dependency between state change v_{11} of TV S_1 and state change v_{21} of TV S_2 is given in the shown TSCBN

  - **State node** v_{ik} indicates the state s_{ij} a TV changes to and has possible states of the TV as outcomes.
  - **Temporal node** Δt_{ik} defines the relative temporal distance of the state change to its latest preceding causally dependent state change.

- Each temporal node Δt_{ik} depends on the state v_{ik} a TV changes to and thus, has an edge

$$(v_{ik}, \Delta t_{ik}) \in E, \forall i \in [1, m], \forall k \leq 0 \tag{1}$$

- If a state change $n_{iq} = (v_{iq}, \Delta t_{iq})$ causally depends on a state change n_{kr} this is represented by edges

$$(v_{iq}, v_{kr}) \in E \tag{2}$$

$$(v_{iq}, \Delta t_{kr}) \in E \tag{3}$$

- **Intra-variable edges** TSCBNs assume that each consecutive state change of the same TV depends on its preceding state. Thus, TSCBNs have the following edges per TV S_i (if m TVs are assumed):

$$(v_{i(k-1)}, v_{ik}) \in E, \forall i \in [1, m], \forall k > 0 \tag{4}$$

$$(v_{i(k-1)}, \Delta t_{ik}) \in E, \forall i \in [1, m], \forall k > 0 \tag{5}$$

– **Inter-variable edges** If a state change $n_{iq} = (v_{iq}, \Delta t_{iq})$ of TV S_i causally depends on a change n_{kr} of another TV S_k representing edges are

$$(v_{iq}, v_{kr}) \in E, i \neq k \tag{6}$$

$$(v_{iq}, \Delta t_{kr}) \in E, i \neq k \tag{7}$$

– **Initial nodes** Each TV S_i has two initial nodes $n_{i0} = (v_{i0}, \Delta t_{i0}) = (s_{idef}, 0)$ at time $t^{i0}_{abs} = 0$ in a default state $s_{idef} \in \Xi_i$.

Example In the TSCBN of Fig. 2 and one possible outcome that is shown in Fig. 3 this applies as follows.

– S_{brake} has six nodes, as state changes might occur at three points in time in the modeled process. Those nodes are three state nodes $v_{brake\ 0}, v_{brake\ 1}, v_{brake\ 2}$ and three temporal nodes $\Delta t_{brake\ 0}, \Delta t_{brake\ 1}, \Delta t_{brake\ 2}$ (in Fig. 2 each $(v_{brake\ i}, \Delta t_{brake\ i})$ is represented as one node).
– State node $v_{brake\ 1}$ models the state the TV S_{brake} changes to in the process, which might be either *active* or *inactive*. Figure 3 shows an outcome of the TSCBN in Fig. 2, where $v_{brake\ 0}$ has outcome *inactive* and $v_{brake\ 1}$ has outcome *active*.
– Temporal node $\Delta t_{brake\ 1}$ models the temporal distance to either $v_{brake\ 0}$ or $v_{ctrl\ 1}$. For the outcome in Fig. 3 $v_{ctrl\ 1}$ is closer, thus, $\Delta t_{brake\ 1}$ here models the time between change $v_{ctrl\ 1}$ ($\langle keep, speeddown\rangle$)and change $v_{brake\ 1}$ ($\langle inactive, active\rangle$).
– For the TSCBN in Fig. 2 the outcome in Fig. 3, among others has initial states *off* for S_{acc} or *inactive* for S_{brake}.
– In Fig. 2 and outcome Fig. 3, the distributions at state $v_{brake\ 1}$ and at $\Delta t_{brake\ 1}$ depend on its previous state $v_{brake\ 0} = inactive$ (intra-edge) and the causally connected state of $v_{ctrl\ 1} = speed\ down$ (inter-edge). $\Delta t_{brake\ 1}$ additionally depends on the outcome at $v_{brake\ 1}$.

2.2 Modeling behavior of states in MSSs

As can be seen in Fig. 4 according to our definition all continuous temporal nodes Δt_{ik} are leaf nodes. Thus, when temporal nodes are excluded, the remaining model of a set of all state nodes v_{ik} and its edges can be inspected as a network of discrete RVs (called *state model*). This allows inference algorithms of standard BNs to be applied on the state structure of a TSCBN, e.g. the Most Probable Explanation (MPE) could be determined to identify dominant system behavior. Based on Fig. 4, in the following it is described how TSCBNs can be used to model this type of behavior of MSSs.

2.2.1 Univariate state sequence

First, we discuss how a one dimensional MSS is modeled as TSCBN. For this, we only consider one TV of Fig. 4 at a time, with the respective part of the TSCBN (i.e. nodes v_{1k} for S_1 and nodes v_{2k} for S_2, without inter-edges).

In the outcome shown for S_2 the state changes at each component of the model are fully observed (from D to E to F), i.e. here the observed outcome $X_2 = \langle D, E, F\rangle$ can be directly represented in the three nodes ($v_{20} = D, v_{21} = E, v_{32} = F$).

However, in a different outcome not all state changes might have occurred, e.g. we could have observed $\langle D, F\rangle$. This is the case if we consider the outcome shown for S_1. Here, the outcome $X_1 = \langle A, B, A\rangle$ is observed, while the actual latent state sequence is $\langle A, B, B, A\rangle$, i.e. at the second B causality exists and a state change could have occurred. TSCBNs model such latent occurrences explicitly as nodes, where any outcome combination, e.g. ($v_{10} = A, v_{11} = B, v_{12} = B, v_{13} = A$), directly maps to an unequivocal observation, e.g. $\langle A, B, A\rangle$

During training a challenge arises from this. From the observation $\langle A, B, A\rangle$ the true latent sequence is not clear. Multiple other latent sequences could be assumed from this observation, e.g. $\langle A, A, B, A\rangle$ or $\langle A, B, A, A\rangle$. Thus, when estimating the latent parameters v_{ij} of a TSCBN from multiple observed sequence examples of X_2, latent estimation is required. In Sect. 4 we propose an approach for this.

Running example In the example of Fig. 2, per observation of an adaption process, we get 5 state sequences. For instance, in the case of being too far away from the preceding car the car needs to speed up until the distance is within the target range. This gives $S_{mode} = \langle Sport\rangle$, $S_{acc} = \langle off, high, off\rangle$, $S_{brake} = \langle inactive\rangle$, $S_{ctrl} = \langle keep, speed\ up\rangle$, $S_{dist} = \langle too\ far, ok\rangle$. Here, S_{brake} did not change its state, which is why only the state $inactive$ was observed, while the true latent state sequence is $\langle inactive, inactive, inactive\rangle$. Notably, this latent representation, which models each potential state change as RV, is useful, as the distance S_{dist} does depend on the brake being active or not at the respective state node 1. In another scenario it could have been $active$, which would lead to a growing distance S_{dist} in node 1 of S_{dist}.

2.2.2 Multivariate state sequence

So far only one dimension (i.e. one TV) was considered. If multiple TVs are given, states of TVs may causally depend on states of other TVs. Such causal dependence is defined with an edge between the corresponding state changes, e.g. in Fig. 4 the state of v_{21} the TV S_2 changes to, not only depends on its previous state, but also on the state change v_{11} of TV S_1.

In particular, this amplifies the above challenge of estimating parameters from observed state sequences, as those dependencies need to be included during this multidimensional parameter estimation.

Running example In the example of Fig. 2, the initial value at node 0 of S_{dist} causally determines what the controller S_{ctrl} does at state 1. If it is $too\ high$, S_{ctrl} is likely to be $speed\ up$ in node 1, while if the distance is ok, the controller would remain at $keep$ in node 1.

Notably, this TSCBN representation is a joint probability distribution. Thus, behind any state change there is a likelihood that is conditioned on its input state nodes, e.g. given $S_{ctrl} = keep$ at node 0 and given $S_{dist} = too\ high$, there is a defined state change distribution at node 1 of S_{ctrl} which could be $P(1 = keep|keep, too\ high) = 0.04$, $P(1 = speed\ up|keep, too\ high) = 0.95$, $P(1 = speed\ down|keep, too\ high) = 0.01$. This allows for an effective way to perform evidence based inference on this compact representation of any complex process, e.g. as the $P(1 = speed\ down|keep, too\ high) = 0.01$ was learned from observed instances, it can be easily read off that in 1 per cent of cases the a critical misbehavior might occur, which would be resolved by experts during development. In the same way it can be read off that in most cases a *speed up* was correctly initialized.

2.3 Modeling behavior in time in MSSs

Time is modeled by temporal nodes Δt. Each node Δt is a continuous RV, with a distribution Σ, defined by parameters Θ, that is conditionally dependent on the same parents $Pa(v_{ij})$ as v_{ij} and on v_{ij} itself. As all Δt are conditioned solely on discrete parents. The temporal part of each state change can be conveniently represented by having one continuous distribution at each Δt_{ik}, per outcome combinations of its parents. Formally, this is

$$\Delta t_{ij}|Pa(v_{ij}) \cup \{v_{ij}\} \mapsto \Sigma(\Theta(Pa(v_{ij}) \cup \{v_{ij}\})). \tag{8}$$

In particular, it is assumed that the duration of a state does not influence the consequent state and the duration of a TV, i.e. all Δt are leaf nodes.

Absolute time TSCBNs represent time relatively. Thus, the absolute time t_{abs} of a state change event needs to be determined from its latest parent's absolute time. If the event of state change was not observed at a parent node, the time of a state node is measured relatively to its last occurring TV state change $\bar{Pa}$. This can be expressed as

$$t_{abs}^{ik} = \max_{r,s}(\bar{Pa}(n_{rs}).t_{abs}) + \Delta t \tag{9}$$

where $\bar{Pa}(n_{rs})$ are all parents of v_{rs}, that did occur, $\max_{r,s}(\bar{Pa}(n_{rs}).t_{abs})$ indicates the latest occurring TV node and Δt is the temporal gap from this parent node to the absolute time of the current node.
In TSCBNs each state node v_{ij} is connected to a temporal node Δt_{ij}. If a state node has exactly one predecessor node, this gap is computed relatively to the absolute time of this predecessor nodes' state change, e.g. if v_{12} occurred at time 17 and the outcome of its consequent state change v_{13} is $\Delta t_{13} = 3$, the time at which the state change v_{13} occurred is 20 ($= 17 + 3$).

If a state node has two or more preceding nodes (i.e. parent nodes) the absolute time of this node's state change is determined relatively to the time of the parent nodes' state change that occurred last, e.g. in Fig. 4 the absolute time of the state change v_{21} is determined relatively to the absolute time of its parents v_{11} and v_{20}. If change

v_{11} occurred at time 7, v_{20} at time 0 and $\Delta t_{21} = 4$, the time of v_{21} is determined relatively to v_{11} (as $7 > 0$). The resulting absolute time of the state change v_{21} would be 11 ($= 7 + 4$) in this example. Notably, at worst every temporal node of a TV has a reference point in its initial node, at absolute time $t_{abs}^{i0} = 0$, i.e. $\Delta t_{i0} = 0$.

Absolute time in latent sequences As stated above in a latent sequence nodes may not be observed and thus, may have no absolute time, e.g. in S_1 the time of transition from B to B is not observed. Thus, using this point as reference for succeeding nodes' time of occurrence Δt is not possible. In this case the relative time is measured from the point of the last observed state change. If for a node in a TV no parent occurred, the last observed state change of the corresponding TV is used as reference point, which is at worst the initial state which is well defined. For instance, let's assume the outcome $\langle B, A, B, A\rangle$ for sequence S_1 with observed absolute times of state change $\langle 0, 5, 8, 10\rangle$ and $\langle D, D, F\rangle$ for sequence S_2 with observed times $\langle 0, 7\rangle$. Then, the time of $v_{21} = D$ is not observed. Thus, no absolute time of this nodes is known from the observation, i.e. the absolute time of the state change $v_{22} = F$ cannot be found relatively to v_{21}. Instead this hidden node v_{21} is skipped and the next observed parent v_{20} is used as reference in a TSCBN. Here, $\Delta t_{22} = 7$ (under the given condition of state changes) defines the absolute time of v_{22} to be 7 ($= 0 + 7$).

Running example In the example of Fig. 2, the acceleration S_{acc} at node 1 is conditioned on its previous node, on S_{ctrl} node 1 and on S_{mode} node 0. Thus, its temporal gap is stored relative to the time that the $ctrl$ variable changes its state in 1, e.g. the distribution at S_{acc} node 1, given defined outcomes for its parents (e.g. $S_{mode\ 0} = Sport$, $S_{acc\ 0} = off$, $S_{ctrl\ 1} = speed\ up$) could be a Gaussian with mean 0.4 s, which indicates that for this parent state combination, it takes about 0.4 s until the acceleration starts, i.e. $P(\Sigma|(S_{mode\ 0} = Sport, S_{acc\ 0} = off, S_{ctrl\ 1} = speed\ up)) = \mathcal{N}(\mu = 0.4s, \sigma^2)$. Notably, other parent conditions might have led to different timings. This, shows that the model capacity in terms of timing is very specific, as it is stored per possible constellation of its causal influences.

2.4 Compact representation

In practice it is often required to discuss such models with experts (e.g. for specification mining as described in Sect. 6). For this and for the purpose of a simpler visualization the TSCBN can be written in a more compact manner. Each state change v_{ik} and its time of change Δt_{ik} can be condensed to one node $n_{ik} = (v_{ik}, \Delta t_{ik})$. Also, connecting edges between two nodes v transform to connecting edges between the corresponding nodes n. Edges to temporal nodes Δt are implicitly assumed, e.g. the model of Fig. 4 can be condensed to the model shown in Fig. 5. The running example in Fig. 2 uses this representation, as well.

2.5 Discussion

The limitations of existing approaches for modeling of MSSs and how those are overcome by TSCBNs, are laid out next, using the scenario of Fig. 3. Extended details of this discussion are provided in Appendix A.

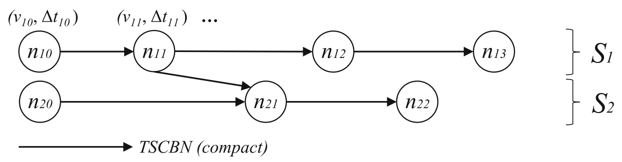


Fig. 5 The compact representation of a TSCBN defines each node n as the state a TV S_i changes to and its time of change

Compactness and parameter complexity Compact representations are desirable, as those require to train less parameters and are easier interpretable. Process models are complex in size, as nodes represent TVs with outcomes. Time-sliced BNs repeat a static structure. Among interval-based approaches TBNEs (Arroyo-Figueroa et al. 2005) are comparably compact, while containing less temporal information. Other approaches are qualitative (Aliferis et al. 1996)) or use trees that explode with process lengths (Galán et al. 2002). CTBNs store more variants yielding a high parameter space.

TSCBNs model state changes where change is possible, making it highly compact and well interpretable as temporal patterns can be directly read of, e.g. in Fig. 3 one can directly read that S_{dist} causes changes in S_{ctrl}, that in turn influences both the acceleration S_{acc} and braking S_{brake} actions. No other process model is able to provide this degree of expressiveness for MSSs.

Explicit modeling Existing approaches can model MSSs' processes either explicitly (e.g. Process Models) or implicitly [e.g. tree-like BNs (Galán et al. 2000)]. The further, is advantageous when analyzing behavioral patterns in data, while the latter is mostly suited to answer specific types of questions (e.g. diagnose a disease given symptoms in temporal order). Existing explicit models include TBNEs, Process Models or unrolled DBNs. TBNEs are less expressive due to to continuous nodes, Process Models do have high numbers of edges and repetition in DBNs does not represent the actual process flow. TSCBNs are the first BNs that provide an expressive unrolled representation of the whole modeled process in multiple dimensions in state and time, which enables effective multidimensional inference, e.g. by computing JPDs new types of analyses are enabled, such as finding of rare state changes.

Continuous temporal modeling Representing time continuously allows for better representation of temporal behavior. In TSCBNs timings can be directly analyzed by setting evidences on the TSCBN and by inspecting the distributions at each node. This is less precise in DBNs and CTBNs as timings are represented through discretization and statically conditioned matrices respectively. Process models are similarly able to determine such timings through path analysis, while this is more complex, then reading it from a TSCBN structure.

3 TrieDiscover: novel structure discovery with tries

To allow for inference the structure of TSCBNs needs to be discovered from a set of observed MSSs. This, requires an automatic discovery approach. Existing approaches do not allow to efficiently discover an interpretable structure for TSCBNs, which is why in this section a novel approach is presented. This is challenging, as possible edge combinations explode with more state changes, data is incomplete, outlier MSSs are possible and temporal aspects need to be included to find plausible edges.

Running example In Fig. 2, we might want to learn how the adaption system of the cruise control works. This includes what causal influences there are between TVs, as well as how likely certain behavior is. For this, we observe and record this process multiple times in the shape of a set of MSSs, e.g. $\{\langle S_{mode} : Sport, S_{acc} : off, S_{brake} : inactive, S_{ctrl} : keep, S_{dist} : too\ far, S_{ctrl} : speed\ up, S_{acc} : high, S_{dist} : ok, S_{ctrl} : keep, S_{acc} : off\rangle, \langle ...\rangle\}$. Another possible MSS is shown in Fig. 3. With a set of MSSs of this shape the structure shown in the figure can be learned, with the method proposed in this section.

3.1 Related work: existing approaches

Structure Discovery (SD) approaches in BNs are typically categorized in score-based, constraint-based and hybrid approaches. When it comes to discovery of dynamic structures, we further use the categories of learning of temporal BNs and process mining, which is a related field for learning process models from sets of sequences.

The learning of structures of BNs is a challenging problem due to the exponential increase of the space of potential network structures with the number of nodes. Most approaches reduce complexity by constraining the space of allowed structures, specifying orders of variables, applying temporal conditions, including prior knowledge or limiting the network size.

Score-based Such approaches start with an initial graph, evaluate the graph using a score function and try to ultimately optimize the structure to maximize the function's score. Performance of those methods depends on the choice of reasonable scoring functions, well constraining the space of allowed structures, by designing an appropriate optimizer and by encoding of the networks. Common scores include the Bayesian Information Criterion (BIC) (Schwarz et al. 1978). Classical approaches include the K2 algorithm (Cooper and Herskovits 1992) or Greedy Hill Climbing (GHC) (Heckerman et al. 1995). Those approaches do not guarantee to converge to the global optimum. For global optimization this was solved in several ways. Those include among many others formulating a linear optimization problem and solving it with branch and cut (Bartlett and Cussens 2013) or constraint programming. Score-based approaches work well for less data, but suffer from bad computational performance for growing data. In addition to that, directions of edges are not clear which is especially relevant when modeling temporal data.

Constraint-based Such approaches constrain potential edges and use Conditional Independence (CI) tests to find the network structure. Typical CI tests include the χ^2 or G tests. Traditional approaches include the SGS approach (Spirtes et al. 2000) or the PC algorithm (Spirtes and Glymour 1991), which use CI tests on subsets of Random Variables (RV) to discover structures. Possible constraints include structural constraints given by properties of the specific objective network (De Campos et al. 2003) (e.g. Markovian assumption in DBNs) or heuristics to limit potential parents of nodes (Scanagatta et al. 2015). This type of methods tends to find hidden common causes, handle selection bias and work well with sparse graphs (Daly et al. 2011).

Hybrid Other works connect score and constraint-based approaches to combine their best properties. This was done by CI tests to find initial ordering of RVs or initial graph skeletons, which are used as input for consequent optimization (Tsamardinos et al. 2006).

Temporal BN learning When learning static DBNs the Markovian assumption is assumed and the discovery is decoposed into learning edges within a time slice and between subsequent time-slices. Murphy and Russell (2002) the further is done with above approaches as in the static case, while the latter reduces to a feature selection problem where each node chooses one or more parents from the previous time-slices. In further works, SD approaches for DBNs with no Markovian assumption (Tucker 2001) or stationarity assumption (Song et al. 2009) were presented. Further approaches, that are comparable to our case, use event sequences as input and Process Discovery algorithms to create the network. The approach described in Savickas and Vasilecas (2014) extracts a Directed Acyclic Graph (DAG) from event sequences by adding edges to the BN if two events directly follow each other in multiple event sequences. Then, in case of cycles in the model dedicated edges are removed. However, this approach is not applicable to our scenario, as we aim at modeling different events of the same TV using more than one node. This strategy was improved by the authors in Savickas and Vasilecas (2017). There, a unique label is assigned to each of the events to prevent the formation of cycles in the BN and the labeling of events is done in a naive way. TrieDiscover finds an advanced solution to this problem of uniquely labeling events, in that it allows to handle optionally occurring events, as discussed in Sect. 3.4.
All of the algorithms in this section so far assumed dependencies between events if the events directly follow each other in the sequences. No statistical tests or score optimization is used to check for independencies. This is solved in Sutrisnowati et al. (2013), where the authors use a score-based procedure to filter edges in the dependency graph of the Heuristic Miner using a mutual information score. Nevertheless, this does not solve the problem of connecting events that are not directly consecutive which is required in TSCBNs.

Process discovery In the Process Mining (PM) community multiple algorithms were proposed for mining a process model from event logs (usually extracted from business processes). The algorithms proposed in this field are related to the discovery of temporal BNs. However, those approaches do not model multiple dimensions and TV structure but rather event sequences. Important algorithms of this type that were incrementally improved include the Alpha Miner (Van der Aalst et al. 2004) for discovery of Petri nets. This method did not allow for silent transitions, i.e. optionally occurring events in sequences are not represented. Thus, this was extended in the Heuristic Miner (Weijters et al. 2006) that includes support of edges in terms of occurrence frequencies, but cannot handle the concept of parallel events. Hence, this was improved in the Inductive Miner and Inductive Miner infrequent (Leemans et al. 2013), where sequences and its representations are decomposed according to defined rules that allow to extract multiple temporal concepts (e.g. *parallel*, *subsequent*). Other approaches, such as Wal-

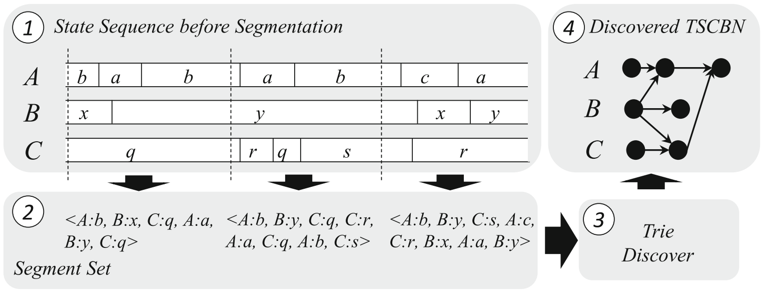

1 State Sequence before Segmentation
A b a b a b c a
B x y x y
C q r q s r
4 Discovered TSCBN
A
B
C
2 <A:b, B:x, C:q, A:a, B:y, C:q>
<A:b, B:y, C:q, C:r, A:a, C:q, A:b, C:s>
<A:b, B:y, C:s, A:c, C:r, B:x, A:a, B:y>
Segment Set
3 Trie Discover

Fig. 6 The basics step of the discovery approach are shown. Starting from a long trace the input MSSs for TrieDiscover are deduced by segmentation. With this TrieDiscover finds a BN structure to represent a set of MSSs

icki and Ferreira (2011) use tries for partitioning, i.e. to assign each sequence element to a process. In contrast to this, our approach assumes one process and uses tries to represent possible paths with those that are merged to get one temporal model.
All of the aforementioned approaches are less effective and efficient for the SD of TSCBNs, as those do not optimally reduce the search space towards multidimensional states, do not focus on searching inter-edges and are not able to handle optionally occurring events. Our novel approach solves those issues. In Sect. 3.4 we discuss differences to, and limitations of existing approaches in more detail. To be able to do this in the following TD is introduced.

3.2 Overview of TrieDiscover

Figure 6 shows the basic steps of the discovery, where multiple MSSs are used as input to learn a representation of those as TSCBN. A pseudocode of TrieDiscover is given in Appendix B.

Basic idea Given a set of MSSs *TrieDiscover* performs two steps for discovery, called *Parent Candidate Identification* and *Structure Optimization*. The further step reduces the search space, while the second step tests for causal edges.
In the first step sequences of TVs are considered, i.e. sequences $\hat{M}$ of TVs rather than MSSs. For this the MSS $M = \langle E_1, ...E_l\rangle$ is transformed to a TV sequence $\hat{M} = \langle S_i, ...\rangle i \in [1, m]$, with m as number of dimensions. For instance in Fig. 3 this could be $\langle S_{mode}, S_{acc}, S_{brake}, S_{ctrl}, S_{dist}, S_{ctrl}, S_{acc}, S_{dist}, S_{ctrl}, S_{acc}\rangle$. The set of all observed sequences is denoted as $\mathcal{M} = \{\hat{M}_1, \hat{M}_2, ...\}$.
Further, we assume intra-edges implicitly. With this, the objective of *TrieDiscover* is to determine the optimal set of nodes and inter-edges.

Example For better comprehensibility we use the following set of TV sequences with TVs $\mathcal{S} = \{A, B, C, D, E, F, G\}$ as an example throughout this section:

$$\mathcal{M}_{obs} = \{\langle A, B, A, D, A, E, G\rangle, \langle A, C, A, D, A, F, G\rangle, \\ \langle A, B, D, A, E, G\rangle, \langle A, C, D, A, F, G\rangle\}$$

Fig. 7 Trie modeling the observation of four MSSs

$$\begin{tikzcd}
& & A \arrow[r] & D \arrow[r] & A \arrow[r] & E \arrow[r] & G \\
& B \arrow[ur] \arrow[dr] & & & & & \\
& & D \arrow[r] & A \arrow[r] & E \arrow[r] & G & \\
A \arrow[uur] \arrow[ddr] & & & & & & \\
& & A \arrow[r] & D \arrow[r] & A \arrow[r] & F \arrow[r] & G \\
& C \arrow[ur] \arrow[dr] & & & & & \\
& & D \arrow[r] & A \arrow[r] & F \arrow[r] & G &
\end{tikzcd}$$

Fig. 8 DAWG after minimizing the trie in Fig. 7

$$\begin{tikzcd}
& & A \arrow[dr] & & & & \\
& B \arrow[ur] \arrow[rr] & & D \arrow[r] & A \arrow[r] & E \arrow[dr] & \\
A \arrow[ur] \arrow[ddr] & & & & & & G \\
& & A \arrow[dr] & & & & \\
& C \arrow[ur] \arrow[rr] & & D \arrow[r] & A \arrow[r] & F \arrow[uur] &
\end{tikzcd}$$

Notably, two sequences start with AB and end with EG, two sequences start with AC and end with FG. Also, the second occurrence of A is optional, as it occurs in half of the sequences only.

3.3 Detailed description of TrieDiscover

1. **Trie creation** We use *Tries* (Bodon 2005) to obtain a compressed representation of all sequences $\mathcal{M}_{obs}$, e.g. yielding the trie in Fig. 7 from $\mathcal{M}_{obs}$.
   Filtering By counting the support of each path within the trie, edge weights are assigned as the number of times an edge was observed in $\mathcal{M}_{obs}$. An outgoing edge e of node X is filtered out if it was observed less than k times the total number of observations of node X (if $frequency(e) < k \cdot frequency(\mathrm{X})$).
2. **Minimization through subtree merging** The created trie resembles a lossless compression of MSSs. This representation has many disjoint branches of potentially overlapping sub-paths. Therefore, identical sub-paths are merged to get a more expressive representation, called a *directed acyclic word graph* (DAWG), as shown in Fig. 8.
   Merging approach Merging of branches is performed using the method proposed in Bubenzer (2011) from automata theory. For this we convert the trie to a deterministic finite automaton (DFA) (trie edges are states, nodes are state transitions). The approach performs a post-order depth-first tree traversal. Each node is assigned a sub-tree code describing the nodes' successors. If two nodes are assigned the same sub-tree code, the nodes are merged, which yields maximal lossless compression. In our example this gives the structure in Fig. 8.
3. **Node indexing** The minimal DAWG is used to assign occurrence indices to all nodes. This is done by first, traversing the graph in topological (= Index Assignment) and then, in reversed (= Index Refinement) topological order.
   Index assignment The first traversal assigns provisional indices to the nodes. Starting from the root node with { A_0, B_0, C_0, D_0, E_0, F_0, G_0 }, the set of previous nodes (parents set) is recursively passed to each node in the DAWG. Each

$$\begin{tikzcd}
 & B_1 \arrow[r] & C_1 \arrow[r] & B_2 \arrow[r] & D_1 \arrow[dr] & & \\
A_1 \arrow[ur] \arrow[dr] & & & & & F_1 \arrow[r] & B_3 \\
 & C_1 \arrow[rr] & & B_? \arrow[r] & E_1 \arrow[ur] & &
\end{tikzcd}$$

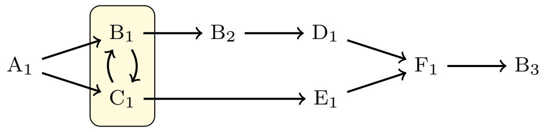
Fig. 9 DAWG that contains an event with non-unique assignment to temporal nodes

node is indexed based on the highest index of the matching TV in the parents set. For example, the root node in Fig. 8 is indexed as A_1 since the latest state change of TV A in the passed set is A_0. If a node has more than one predecessor (e.g. node G in Fig. 8) all parents sets are merged before an index is assigned.

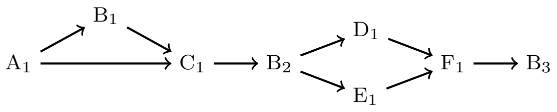
In the example this gives us $\langle A_1, B_1, A_2, D_1, A_3, E_1, G_1\rangle$, $\langle A_1, C_1, A_2, D_1, A_3, F_1, G_1\rangle$, $\langle A_1, B_1, D_1, A_3, E_1, G_1\rangle$ and $\langle A_1, C_1, D_1, A_3, F_1, G_1\rangle$. Notably, this approach correctly identified that the second occurrence of TV A in the latter two sequences corresponds to the node A_3 rather than node A_2 as a naive assignment would assume. Passed parents sets at each node resemble candidate inter-edges to the current node.

Index refinement The first step might yield cases in which the index is not unique in the sense that there is an occurrence gap, e.g. in Fig. 9 the lower path from A_1 to B_3 contains one occurrence of TV B, which can be either B_1 or B_2. Such ambiguous occurrences are refined in a second traversal in reverse topological order. In Fig. 9 B was provisionally indexed as B_1 in the first traversal. By passing a successors set backwards the second traversal identifies missing occurrences, such as the missing assignment of B_2 on the lower path. Determination of which assignment is preferred is done using the number of strongly connected components (SCC) when merging nodes of the same TV and index. More SCCs are preferred. Figure 10a shows the resulting merged graph when event B in the lower path of the graph of Fig. 9 is assigned to node B_1. There, it contains seven SCCs as the nodes B_1 and C_1 form one SCC. When assigning it to B_2 the graph in Fig. 10b yields eight SCCs and thus, is preferred.

4. **Parent candidate identification**
   Union of parents sets After the previous step some nodes in the DAWG might belong to the same TV and index (e.g. nodes D in the DAWG in Fig. 8), but have different parents sets. In Fig. 8 parents of D are B_1 or C_1. Both parents are assumed candidate parents here. In case that two nodes X_i and Y_j occur in parallel (i.e. sometimes X_i precedes Y_j and sometimes Y_j precedes X_i), both nodes contain one another in their parents set. We assume that in this case nodes are independent. Thus, nodes are removed from each other's parents set. Such nodes can be identified by SCCs with more than one node (as it is the case in Fig. 10a).
   Temporal filtering of parents sets Complexity is further reduced by filtering of parent candidates by defining a temporal threshold t_{th}. As all state changes of TVs include timestamps relative to the beginning of the sequence, averaging of relative times between individual nodes can be used to determine average time delay between two arbitrary nodes. Parents that are further than t_{th} away from a node are removed from the node's parents set.
5. **Structure optimization** From the identified parent candidates per node of the TSCBN, a set of optimal parents is found through structure optimization, which



(a) Resulting merged graph when the problematic event is assigned to temporal node B 1. Resulting SCC is highlighted in yellow.



(b) Resulting merged graph when the problematic event is assigned to temporal node B 2.

Fig. 10 Resulting merged graphs for two different event assignments. The assignment in Fig. 10b leads to one more SCC and thus is preferred

can be done either using score-based (SB) optimization procedures or CI tests. Both approaches are applied here. When using SB optimization, we call our approach *Score-based (SB) TrieDiscover* and when using CI tests we call it *Constraint-based (CB) TrieDiscover*. SBTrieDiscover uses a decomposable score (e.g. BIC, AIC, K2, BDeu) for optimization, while CBTrieDiscover uses CI tests to find optimal parents per node.

However, performing CI tests can be computationally expensive as the Markov Blanket with various condition sets needs to be iterated. Therefore, in a third approach we perform a χ^2 test to filter out connections between RVs (i.e. nodes of TVs) that are below a correlation threshold χ_{th}. The remaining candidates are reduced with CI tests. For the CI tests edges with a significance level lower than a threshold α are removed.

Conversion to TSCBN Intra-edges are defined inherently by connecting nodes that correspond to the same TV, e.g. B_0, B_1 and B_2. Inter-edges for each node of a TV are defined by the parent set of that node that remains after the structure optimization step. Notably, any event in the tree starts at index 1 for each TV (e.g. B_1). Thus, as a corresponding parent with index 0 is passed to those nodes as candidate set, initial nodes are always present with TrieDiscover (e.g. B_0).

Complexity The complexity of TrieDiscover is deduced in Appendix B. Given the number of sequences m and the average length L of sequences $\hat{M}$, the complexity of all steps prior to structure optimization (SO) is $\mathcal{O}(m \cdot L)$. SO uses existing methods which determine the additional complexity. At worst every node could have the respective maximally possible number of parents in its parent set, yielding no benefit of TD. At best, the optimal set of parents is found per node. Using for instance score-based SO, this gives complexity $\mathcal{O}(m \cdot L \cdot p)$, where p is the average parent set size. If p is limited this gives $\mathcal{O}(m \cdot L)$.

3.4 Discussion

TrieDiscover directly discovers structures of TSCBNs. This requires to exploit temporal constraints, to capture optionally occurring events and to focus on the search of inter-edges rather then, intra-edges. Existing approaches add overhead in either of these categories, as those focus on the general scenario of events rather then, multidimensional state sequences. TD is the first method, to combine these aspects to yield an efficient discovery and an effective representation.

Reduced search space by temporal constraints TD can be categorized as a hybrid approach. For the given scenario, this is preferred over score and constraint-based methods. The further explode in complexity due to a high number of combinations that are possible. The latter, do not inherently include temporal aspects. Further, Process Mining methods assume one dimensional sequences and do not include multidimensional and statistical relations of the data.
By reducing the search space through heuristic temporal constraints (e.g. temporal range) first, and then, performing CI tests on remaining plausible edges both temporal, multidimensional and statistical relations of the data are captured. This is achieved by using approaches similar to Process Discovery in the first phase. Unlike any other approach TD is the first to perform the further on state changes rather than events.

Search for inter-edges Comparable approaches do assume the general scenario of events that might be arbitrarily interlinked. In contrast to this, TSCBNs do not look for edges among nodes of the same TV, but for inter-edges only.

Smart node indexing TrieDiscover is the first SD algorithm of BNs to not use a naive node indexing. As stated before, comparable methods (Savickas and Vasilecas 2014) add nodes per directly consecutive event and remove edges if cycles occur. This is not applicable here, as modeling of multiple events of the same TV is required. Savickas and Vasilecas (2017) this is solved by using unique index per event that prevent formation of cycles. However, indexing of events is done in a naive way. That is, all the ith occurrences of an event of TV X are indexed X_i. The algorithm we present in this work is the first approach, to find an advanced solution to this problem. This is achieved by allowing to handle optionally occurring nodes, e.g. in sequences $\langle B, A, A, B \rangle$ and $\langle B, A, B \rangle$, in the first sequence labeling of A might be A_1 or A_2. To discover the best structure here, an effective traversal strategy and inspection of SCCs is used.

Handling noise TD is well able to handle noise by using a technique adapted from Process Mining, i.e. TD uses early filtering to focus on relevant edges only. This is especially relevant when applied to real world data, where automated segmentation may yield imperfect sequences. This effect is also exemplified in the case study in Sect. 6. Comparable approaches that do not apply such constraints contain additional redundancy and computational overhead, as combinations of edges of rare paths are included in computation which adds to the structural search space.

4 Parameter estimation

Given the structure, TSCBNs (e.g. in Fig. 2) are parameterized with a set of MSSs. For this, the same MSSs are used as during structure discovery. However, to learn temporal behavior, additionally the timings are given, which are used to learn parameters of temporal nodes, e.g. in the running example a MSS set might be $\{\langle(t = 0.0, S_{mode}$:

$Sport), (t = 0.0, S_{acc} : off), (t = 0.0, S_{brake} : inactive), (t = 0.0, S_{ctrl} : keep), (t = 0.0, S_{dist} : too\ far), (t = 0.1, S_{ctrl} : speed\ up), (t = 0.5, S_{acc} : high), (t = 2.4, S_{dist} : ok), (t = 2.5, S_{ctrl} : keep), (t = 2.9, S_{acc} : off)\rangle, \langle ...\}$.

TSCBN estimators For fully observed MSS (i.e. at every node there are state changes) the TSCBN model can be estimated using non-latent approaches such as MLE and Bayesian Estimation with Dirichlet priors (details given in Appendix C). However, when dealing with latent MSSs, per TV, nodes are only partially observed and thus, latent estimation methods are required.
For this we designed two approaches, which are an EM approach and a Coordinate Ascent Variational Inference (CAVI) approach. In contrast to existing estimation approaches, our estimator needs to tackle the following challenges. First, a latent observation of one TV (e.g. for TV S_1, $\langle A, B\rangle$) correlates to a subset of nodes (e.g. n_{11}, n_{12}, n_{13}) in the TSCBN, where nodes of subsets might be interdependent (e.g. by connected inter-edges, n_{12} and n_{23}). Second, the mapping per observation to possible latent estimates is constrained in time, i.e. temporal order of events needs to conform with nodes and directions of edges in the TSCBN. Also, in state, per TV only a combinatorial subset of outcomes is possible (e.g. $\langle A, B\rangle$ can be $\langle A, A, B\rangle$ or $\langle A, B, B\rangle$).
We solve this in EM by sampling from TSCBNs and dropping inconsistent samples and in VI by deducing a set of update equations, by finding a meaningful computation per Δt and by excluding invalid mappings from full to latent TV sequences during computation of expectations. Further, we introduce a naive MLE approach, where similar sampling as in EM is used to approximate samples from latent to full observations. The full observations can then be directly computed via MLE of the TSCBN.

Formalization Each latent sequence of a TV S_i is defined as a latent variable $Z_i = (v_{i1}, v_{i2}, ...)$ and the observed sequence as X_i. The set of all latent variables for observation k is $Z^k = \{Z_1^k, Z_2^k, ..., Z_m^k\}$ and the set of all observed sequences is $X^k = \{X_1^k, X_2^k, ..., X_m^k\}$. $X = \{X^1, X^2, ...\}$ are all observations and $Z = \{Z^1, Z^2, ...\}$ the according latent sequences. For the latent case all Z^k are assumed missing at random. Further, we define the empirical distribution, that results from the observations Z as $q(v_{ij}|Pa(v_{ij}))$.

4.1 Random maximum likelihood estimation

When latent variables are not fully observed, estimates need to be found to approximate the parameters of the TSCBN. One approach to do this is using a standard sampling based MLE approach (MLE-R) which is presented here. We exploit the convenience, that continuous TVs are leaf nodes in a TSCBN and thus, can parameterize all temporal nodes and all state nodes separately as follows.

Temporal node estimation As temporal nodes depend on its corresponding state node, as well as on this nodes' parents, one distribution needs to be found per condition combination. For this, first, per node Δt and per condition, a distribution is defined (e.g. Gaussian). Second, during the MCMC sampling procedure all n_{samp} gaps for Δt are recorded and stored under its respective parent conditions. Then, per condition set at each node Δt, the recorded gaps are used to fit a distribution over the observed gaps and the corresponding distribution parameters are used in the temporal nodes.

State node estimation A common way to handle latency is by meaningfully imputing missing values (Van den Broeck et al. 2015) at each observation and then applying non-latent approaches (e.g. MLE) for parameter estimation, as follows.

1. Per input sequence X_i^k draw a valid random mapping from the current TSCBN to get Z_i^k. This results in a list of full observations Z (in state and time).
2. Use Z to perform MLE to find parameters of the TSCBN.
3. Repeat step 1) and 2) until convergence, with the newly updated parameters that were found in 2).

A main disadvantage of this approach is that combinations of outcomes for the TSCBN that were not drawn during the sampling process do not update the corresponding parameters. Thus, those parameters remain in its initial state, which might e.g. be random or uniform. This is overcome in latent variable estimators such as EM and VI.

4.2 Expectation maximization

In EM the goal is to maximize the likelihood $L(\Theta|X, Z)$ of the TSCBN given the observed data. This means, it maximizes the expectation of *(i)* the TSCBN with parameters Θ to produce all latent sequences Z and *(ii)* Z and Θ to produce the observed sequences X. Formally, this means EM maximizes

$$\Theta^* = \arg\max \mathbb{E}_{Z\sim p(Z|X,\Theta)}[\log p(X, Z|\Theta)]. \tag{10}$$

EM approach To compute the expectation per TV exactly all combinations of mappings from X_i to Z_i would need to be computed. Per sequence S_i with n latent sequence elements and k observed sequence elements, $\binom{n-1}{k-1}$ combinations are possible, which is worst for $\binom{2p}{p}$ or $n = 2k-1$. Thus, e.g. at worst a TV with $n = 20$ has already 184756 combinations, which for growing numbers of n and more TVs becomes in-feasible to compute. Thus, efficient exact computation of $\mathbb{E}_{Z\sim p(Z|X,\Theta)}[\log p(X, Z|\Theta)]$ is not possible.

To solve this, we use a Monte Carlo sampling approximation to determine approximate distributions at each step.

Using K observations and M sequences (i.e. TVs) EM is performed as follows. Per observation k and sequence i a local estimate $q(Z_i^k)$ of the latent sequence that can be mapped from the corresponding observed sequence X_i^k is assumed (e.g. given $X_i^k =$ ABC and 4 nodes for TV S_i, Z_i^k may be ABBC, AABC or ABCC), where Θ holds the parameters of the model at the current iteration.

At first both $q(Z_i^k)$ and all Θ are assumed uniformly distributed, before being updated on each iteration. For this, per observation MCMC sampling from all M $q(Z_i^k)$s is used to draw n_{samp} samples from the whole TSCBN. This gives n_{samp} (or less, as samples not satisfying temporal constraints are dropped) valid outcomes per local latent sequence Z_i^k. With this the following update steps are performed.

Update $q(Z_i^k)$: To maximize the likelihood of the local estimate, the updated estimate $q^*(Z_i^k)$ is computed as $q^*(Z_i^k) = \sum_{t=1}^{n_{samp}} p(Z_{it}^k|\Theta)$, where Z_{it}^k is the tth sample drawn (e.g. $\langle A, A, B, C\rangle$) and $p(Z_{it}^k|\Theta)$ is the likelihood of this sequence to occur.
Update Θ: To update Θ, $\mathbb{E}[\log p(X, Z|\Theta)]$ is found by simply counting all occurrences of all n_{samp} drawn samples under the respective conditioning observed in the sample. With this the likelihood is iteratively maximized.

Temporal constraints To improve accuracy of the EM algorithm (when estimating the latent Z^ks from its X^k) we propose to exclude latent sequence combinations from X^k to Z^k which are not possible according to the given temporal structure of the TSCBN. Any drawn MSSs Z^k is valid if the following is given:

- If a sequence element v_{ik} has an identical state as its previous intra-dependent node ($v_{ik} = v_{i(k-1)}$), the end-time $t_{abs}^{i(k+1)}$ of its interval needs to happen after all its parents' v_{rt} start times t_{abs}^{rt}: $t_{abs}^{i(k+1)} > t_{abs}^{rt}$.
- For all parents v_{rt} with $v_{rt} = v_{r(t-1)}$ of a sequence element v_{ik} with $v_{ik} \neq v_{i(k-1)}$, v_{rt}'s start times need to occur before v_{ik} ends. That is, $t_{abs}^{i(k+1)} > t_{abs}^{rt}$.
- For all parents v_{rt} with $v_{rt} \neq v_{r(t-1)}$ of nodes v_{ik} with $v_{ik} \neq v_{i(k-1)}$, v_{rt} we require $t_{abs}^{ik} < t_{abs}^{rt}$.

4.3 Variational inference

Another common approximation approach for latent variable models, such as TSCBNs is Variational Inference, which we introduce here. We oriented this approach around the framework presented in Bishop et al. (2003), where the general idea of using the Markov Blanket for local computations of the updates in VI in BNs is introduced. We extend this approach for our purpose by *(i)* defining the correlations of an observation per TV as vector that correlates to its latent nodes of the same TV in state and time, *(ii)* adapting computation of expectation to allow only valid mappings (which reduces computational complexity) *(iii)* deducing update equations between continuous and discrete nodes, *(iv)* by including the observed absolute times as Δt values for computation of the expectation with an interpolation strategy.

4.3.1 Coordinate ascent variational inference in TSCBNs

For the VI approach, we assume the probabilistic structure shown in Fig. 4, but with inverted edges reaching from v_{ij} to X_i for all TVs. From this representation, we will deduce update equations to find estimates for all RVs v_{ij} and Δt_{ij}. The result of the update equations is presented in the following, while a detailed deduction of formulas can be found in Appendix C.
In Coordinate Ascent Variational Inference each variational distribution $q^*(z_{ij})$ is updated until convergence using

$$q^*(z_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log p(Z, X)]\right) \tag{11}$$

Assumptions

- $\mathbb{E}$ is both computed over K observations and for all valid latent mappings from X_i to Z_i. Computationally this means to compute $\mathbb{E}$ over valid combinations only.
- According to Bishop et al. (2003) the expectation needs to only be computed over the Markov Blanket around the updated node z_{ij}.

Using those assumptions we deduce update equations $q^*(z_{ij}^k)$ for state nodes v_{ij} and for temporal nodes Δt_{ij}. This is done by iteratively alternating between updating network parameters $p(Z|X)$ from all $q^*(z_{ij})$ and updating $q^*(z_{ij})$ from the current network parameters $p(Z|X)$. This is done as follows.

Update state nodes v_{ij}: With above assumptions and the definition of the TSCBN structure the update equation of any state node v_{ij} is given as

$$q^*(v_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log p(Z, X)]\right) \tag{12}$$

$$= \tag{13}$$

$$q^*(v_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log\left(\Psi(v_{ij}) \cdot \Gamma(v_{ij}) \cdot \Xi(v_{ij})\right)]\right) \tag{14}$$

with Ψ as the factor with all parents of v_{ij}, Γ as all factors that contain co-parents of v_{ij} and Ξ as factor with children of v_{ij}.
Notably, $\Xi(v_{ij})$ includes the observed sequence X_i with factor $p(X_i|v_{ij} \cup CoPa(v_{ij}))$, which is zero for invalid mappings.
Notably, in CAVI the expectations are computed with respect to the update node, i.e. $\mathbb{E}_{q_{-ij}}$ is excluding $q(v_{ij})$.

Updating network parameters: With the local updated estimates the network parameters can be found with

$$p(v_{ij}|Pa(v_{ij})) = mean\left(\prod_{\tau \in \{v_{ij}\} \cup Pa(v_{ij})} q(\tau)\right) \tag{15}$$

With, *mean* as the mean over all K observations. Here, the mean field assumption is applied assuming all nodes $q(v_{ij})$ to be pairwise independent.

Update temporal nodes Δt_{ij}: Here, for nodes Δt_{ij}, we still consider the structure in Fig. 4 and a Gaussian distribution per temporal node. With this, the update equation for the local estimate of Δt_{ij}, governed by its parameters μ and σ per observation is

$$q^*(\Delta t_{ij}; \mu_{ij}^*, \sigma_{ij}^{*2}) \propto \exp\left(\mathbb{E}_{\Delta t_{-ij}}[\log p(Z, X)]\right) \tag{16}$$

For this case again the structure of TSCBNs is assumed, only valid mappings are assumed and only the Markov Blanket is considered per node. This results in the following update equation for the local estimate μ_{ij}^* of the temporal node Δt_{ij}

$$\mu_{ij}^* = \frac{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \mu_{ij|\omega} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)} \tag{17}$$

This makes intuitively sense, as the μ_{ij} governing Δt_{ij} is a weighted average over its outcomes depending on its likelihoods.
Updating network parameters: Finally the estimate $p(\Delta t_{ij}; \mu_{ij|\omega}, \sigma_{ij}^2 | Pa(\Delta t_{ij})$ of each CPD of each temporal node can be found as the weighted average

$$\mu_{ij|\omega}^{*} = \frac{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \mu_{ij}^{*} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)} \tag{18}$$

Estimation algorithm By altering computation of local and global parameters until convergence updates are computed. A pseudo code is provided in Appendix C.

4.3.2 Computing $\boldsymbol{\Delta t}$

When computing any Δt_{ij} in a latent sequence (e.g. $\langle A, A, B, B \rangle$) from a full observation X_i (e.g. $\langle A, B \rangle$), in order to be usable in the computation of the expectations in CAVI each latent node requires a defined absolute time at which it occurs. We use the following interpolation strategy for this.

- Times between two elements are interpolated linearly within known parts, e.g. given A at $t = 1$ and B at $t = 2$ would result in times and values $\langle (A, 1), (A, 1.33), (A, 1.66), (B, 2.0) \rangle$ for a latent sequence $\langle A, A, A, B \rangle$.
- If the last observed element is followed by further elements in the latent sequence, its preceding element's distance is used for interpolation, e.g. having seen A at $t = 1$ and B at $t = 2$ that is extended to $\langle A, A, B, B, B \rangle$ would result in $\langle (A, 1), (A, 1.5), (B, 2), (B, 2.5), (B, 3.0) \rangle$. This extension time is determined by subtracting the last observed time by its predecessors interpolated time. If no predecessor is available the distance is computed by dividing the total time of the current sequence by its number of nodes. While this interpolation may potentially lead to temporal inconsistency, it produces a reasonable approximation that tends to decrease deviation from the true mean when enough sequences were observed.

Running example At this point, each node in the example of Fig. 2 has conditional probability distributions associated in state and time. Dependent on its preceding parent nodes, those nodes determine how likely a certain state change is to occur and at what time it will occur. Those distribution represent the behavior observed from multiple MSSs (here multiple runs of adaption of the cruise control) and thus, reduces the complexity of a set of MSSs to a compact TSCBN that represents this process in multiple dimensions under uncertainty. Using the learned TSCBN, among others, experts are able to determine conditions under which anomalous behavior becomes likely or most prevalent constellations of the adaption (e.g. to use this knowledge to design specifications).

5 Synthetic evaluation

We evaluated our model and the developed parameter estimation against DBNs and CTBNs, as the two main competing approaches. Further, TrieDiscover is compared to multiple baseline SD approaches. Notably, both our code and the data used for evaluation are made available.[1]

5.1 Experimental setup

All Experiments were conducted on an HP^{TM} Z-840 equipped with Intel$^{®}$ Xeon$^{®}$ E5-2640 v3 2.60GHz CPUs and 96 GB of RAM. To compare the models, we performed a synthetic evaluation, where ground truth MSSs are generated that are used for training and evaluation of the models.

5.1.1 Data generator

For synthetic data analysis we implemented a generator for MSSs. This generator is a TSCBN, with defined structure that models evolving dependencies between TVs. Sampling from such a defined TSCBN gives a set of MSSs that is used as input for training of all models. Also, for the comparison of the models the generator TSCBN is used as a ground-truth. The state lengths are sampled from a Gaussian distribution. Additionally, the generator allows to set a defined probability of state change per node. From this model we draw defined numbers of samples used for evaluation. Those samples can be latent sequences per TV (e.g. $\langle S_0 = 0, S_0 = 0, S_0 = 1\rangle$) when drawn, which are reduced to true observations (e.g. $\langle S_0 = 0, S_0 = 1\rangle$).
The parameters that define the MSSs are set by defining the TSCBN structure. This includes the following parameters.

- **Structure of TSCBN** number of TVs n_{TV}, number of nodes per TV n_n, i.e. length of sequence per TV
- **Connections in TSCBN** number of TVs that have a connection with other TVs n_{TVint}, number of edges n_{inter} that nodes have if a connection is present.
- **Parameterization of TSCBN** number of states per state node n_c and settings for μ and σ per temporal node, state change probability p_{SC}, which is the probability with which a node remains in the state it previously was in.

To allow for comparison, we deduce the structure of DBNs and CTBNs from the true structure of the TSCBN. This is done as follows.

1. A TSCBN with n_{TV} TVs and n_n nodes per TV is created. A DBN with n_{TV} nodes per time slice and a CTBN with a fixed structure with n_{TV} nodes are generated.
2. Per TV, n_{TVint} connecting TVs are randomly chosen. Then, n_{inter} connections to nodes of this TV are randomly set. In CTBNs and DBNs those edges form the static network structure. This structure is assumed given for DBNs, CTBNs and TSCBNs when performing parameter estimation. In DBNs the structure is defined by its resolution, which is the breadth of each time slice.

[1] https://github.com/arturmrowca/tscbn.

3. Next, n_c states are created per node and a CPD is randomly generated per node and condition. In TSCBNs, this CPD forms the ground truth and is used for sampling MSSs. Each CPD is created such that all probabilities of a intra-node to remain in its same state are set fixed to p_{SC}, while a random distribution is drawn for the remaining CPD entries. Further, a Gaussian distribution with μ' and σ' is used to draw values μ, that are set to the temporal nodes under given conditions. With this per node different μs are provided and state sequences of equal lengths are avoided. In the experiments σ'^2 was set to 0.1.
4. With this a structure for all three network types is given, while for TSCBNs an additional parameterization is given. The TSCBN is used for sampling sequences as described in the beginning of this section.
5. In all experiments of this section those samples are used as input.

5.1.2 Setup for model and parameter estimation

Experiments After performing steps 1 to 5 of the data generator for the evaluation of the model various variants of TSCBN structures are created. For evaluation of parameter estimations, a copy of the TSCBN is created where all parameters are deleted. This TSCBN is used as input for parameter estimation, while the original is kept as ground truth.
The generated samples are split 90:10 into a training and a test set which are randomly chosen from the data. Then, for all three network types parameter estimation is performed by using the sampled MSSs from the training data set. The estimated parameters are then, used for evaluation by applying the evaluation criteria which are described in the next section. Results of model evaluation are presented in Sect. 5.2, while the results for parameter estimation are given in Sect. 5.3.

Baselines As a benchmark for performance of the model and our parameter estimation we use the two most comparable types of networks, which are DBNs (Dean et al. 1989) and CTBNs (Nodelman et al. 2002). For the further, we used python's libpgm package and for the latter a python wrapper for the R package CTBN-RLE (Shelton et al. 2010) was written and used. In DBNs we define static edges between related TVs. The structure is repeated in discrete time, with outcomes at each slice. The maximum distance of a time-slice to a state change to capture is defined as *DBN tolerance*, which defines the resolution of the structure repetition. Parameter Estimation is done with a Maximum Likelihood Estimator. For CTBNs we define the same static edges between TVs for both the transition and intensity matrices as provided by the ground truth TSCBN. Then, CTBN-RLE's parameter learning engine is used to estimate parameters of the CTBN.

Parameters used For all models we assumed 5 nodes per TV (i.e. 4 intervals), 4 states per node, per TV 2 edges to two other TVs. For the DBN we used a resolution of 0.02 and the length of all intervals is drawn from a Gaussian with $\mu = 0.5$ and $\sigma^2 = 0.1$. The number of TVs n_{TV}, the number of samples for training and test n_{samp}, and the probability of a state change occurring p_{SC} were varied. For parameter estimation per sequence during MCMC sampling 1000 samples are drawn per iteration, 5 iterations were performed for EM and VI. Per iteration a CPD smoothing of $\epsilon = 0.1$ is used.

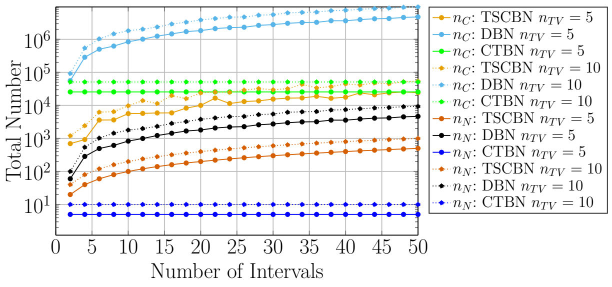


Fig. 11 Results of structural complexity, as number of nodes n_N and CPDs n_C, for various numbers of TVs

5.2 Model

In the following we evaluate the TSCBN model in terms of structure against the existing baseline approaches DBN and CTBN.

5.2.1 Setup

Evaluation criteria We evaluated the models and parameter estimation approaches in terms of structure using the number of edges n_E, nodes n_N, states n_S and parameters n_C. The structure of the models is quantified with the number of components required to model a MSS. For a given model this includes the number of edges n_E, number of nodes n_N, total number of states n_S and the total number of entries required in all CPD tables of all nodes n_C (i.e. the number of parameters).

Experiments To evaluate the model *structure*, the number of intervals and TVs is varied and an according TSCBN, CTBN and DBN found to represent this MSS. The structures are evaluated in terms of number of nodes, edges, states and conditional probability entries (=parameters).

5.2.2 Results

Structural comparison In Fig. 11 the results of the structure evaluation are shown. Both DBNs and TSCBNs need additional nodes n_N per interval, while CTBNs have a fixed number of nodes, which is the number of TVs. DBNs add multiple nodes per state change (depending on its resolution), making it complex in size and in terms of parameters. In contrast to that, TSCBNs require only one node per further interval making it smaller than DBNs. CTBNs are more compact than both approaches in terms of nodes, as those assume a static structure. However, in terms of parameters n_C CTBNs require intensity matrices per condition given in its static network. Thus, especially when multiple TVs are involved that change its dependence on other TVs over time, CTBNs need to store a correlation in its static network per combination of correlated TVs. In contrast to that, in TSCBNs only dependence at its actual time in the process are included, resulting in less edges per node in TSCBNs. Also, CTBNs require to store a matrix per condition and node while TSCBNs only store its CPD and temporal distribution parameters per condition and node. Thus, CTBNs are similarly light as TSCBNs in terms of parameter size, if no evolving dependencies are given, while TSCBNs show to be lighter when evolving dependencies are modeled in MSSs. In our evaluation this becomes evident as we assume two edges to be randomly directed between any TV combination, all of which are modeled in TSCBNs only at the nodes that they occur. In CTBNs all correlations need to be modeled within its TV network structure resulting in more combinations and thus, bigger numbers of parameters.

5.3 Parameter estimation

Here, the performance of the parameter estimation approaches that were proposed in Sect. 4 are evaluated. This is done first, against CTBNs and DBNs when including the model and second, the approaches are compared against each other within TSCBNs.

5.3.1 Setup

Evaluation criteria Evaluation in expressiveness was done in terms of mean log-likelihood and temporal mean log-likelihood, while the performance was evaluated using the runtime.

Mean Log-Likelihood: The mean log-likelihood resembles the state expressiveness of a model computed from N given outcomes with M nodes per outcome. Values closer to zero indicate better expressiveness. It is normalized by the total number of nodes M in the model to allow for fair comparison between TSCBNs, DBNs and CTBN. It is defined as

$$l_{mean}(\Theta|X) = \frac{1}{N \cdot M} \sum\nolimits_{i=0}^{N} \sum\nolimits_{j=0}^{M} \log P(x_i^j|Pa(x_i^j)) \tag{19}$$

where x_i^j is the jth outcome and the ith node of the model. This metric is computed for all discrete nodes x_i^j (i.e. in TSCBNs all nodes v and in DBNs all nodes). The magnitude of this metric is called the Absolute Mean Log-likelihood.

In CTBNs we computed this metric by walking through each sample in a sequence and by storing all previous values of the outcomes of TVs as conditions. Then, per sample the transition matrix per TV is used to find the log-likelihood of the TV transitioning into the observed state under the given previous condition. All computed log-likelihoods are then summed up and normalized by the number of summands yielding the Mean Log-Likelihood. This metric resembles how likely a CTBN would have produced the observed sequence.

Temporal Mean Log-Likelihood: If x_i^j is a temporal node ($x_i^j = \Delta t_i^j$), $P(x_i^j|Pa(x_i^j))$ in $l_{mean}(\Theta|X)$ is the Gaussian of the according temporal node in the TSCBN. It measures the temporal expressiveness of a TSCBN. We call this criterion the temporal log-likelihood in the following.

Experiments The parameter estimation is evaluated by training a model with 90 percent of the data and testing it with 10 percent. During testing the mean log-likelihood of the test sequences are used for evaluation of the trained model. Further, for the

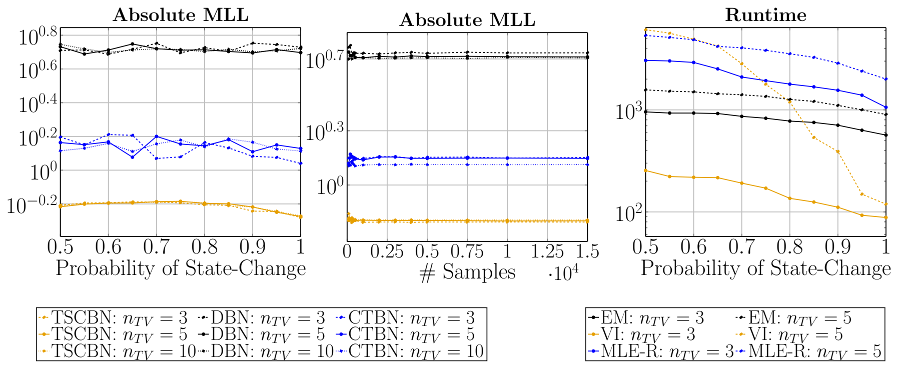


Fig. 12 *Left and Mid*: Abs. MLL and runtime when varying probability of state change and number of training samples using the EM algorithm for estimation of TSCBN parameters. *right:* Runtimes for three approaches including EM, VI and MLE random for two structure sizes

TSCBN the temporal log-likelihood and runtime of the parameter estimation is measured. This is done once with fixed $p_{SC} = 0.8$ and increasing n_{samp} and once with fixed $n_{samp} = 2000$ and increasing p_{SC}.

5.3.2 Results

Comparison in expressiveness Figure 12 shows, that for an increasing number of state changes DBNs have a constant likelihood, while it improves in TSCBNs. This is due to the structure of DBNs, which is independent of state changes and samples data at regular time steps. CTBN are also, less influenced by state changes as CTBNs generalize over those changes (as process independent intensity matrices are learned) and only learn the observed. So, CTBNs ignore structure of events and changes of evolving dependencies, which makes them less expressive. In TSCBNs less state changes require more marginalization during parameter estimation and thus, decrease expressiveness. However, in comparison to CTBNs even for a small state-change probability of 0.55 our approach is about 2 times and for DBNs even 8 times more expressive for equal numbers of TVs when representing MSSs.

Expressiveness of TSCBN Figure 13 shows that parameter estimation in TSCBNs improves with more training samples and smaller state-change probability. This is, as with more training samples more behavior is observed and thus, the model is more likely to expressively represent unknown samples. Higher probability of a state change improves likelihood of the data both in time and state as less marginalization is required. For a small number of samples the temporal log-likelihood does improve significantly if more samples are added, while it stagnates later. Further, it can be seen that less TVs show a better temporal expressiveness, as in this case it is more likely that unknown samples are similar to observed samples in training.

Runtime The runtime improves for less TVs and more expected state-changes that occur which is illustrated in Fig. 12 (right). This is, as sampling is only required for latent sequences, whereas no MCMC sampling is performed if the length of the observed sequence matches the number of nodes for a TV.

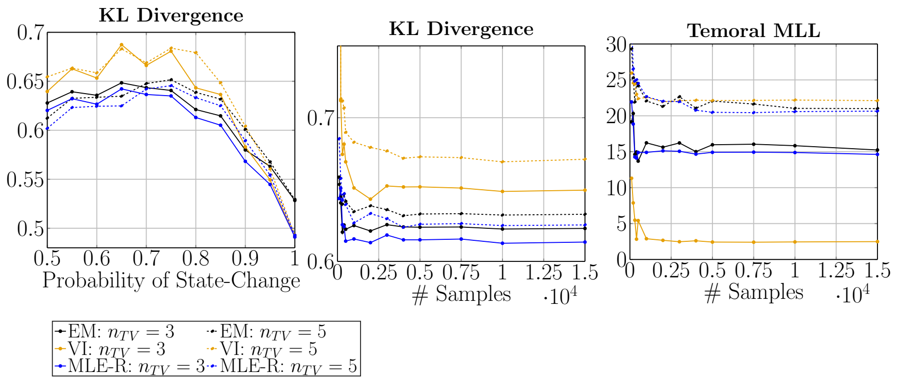


Fig. 13 Results: KL divergence and temp. MLL when varying probability of state change and number of training samples for three approaches including EM, VI and MLE random for two structure sizes

Comparing estimation approaches We also compared the proposed approaches against each other using Figs. 12 (right) and 13. An in depth discussion of these results can be found in Appendix D. Overall, VI performs better in runtime, is slightly less expressive in state, but most expressive in time. MLE-R and EM perform similar in those three categories and yield better expressiveness in state than VI. Consequently, all three approaches are about equally suited for parameter estimation in TSCBNs, while trading off expressiveness in state, expressiveness in time and runtime.

5.4 Structure discovery

In this section we present the results of the evaluation of TrieDiscover in terms of performance and precision.

5.4.1 Setup

We evaluated the presented algorithm TrieDiscover by generating random TSCBNs with the generator described above and by using samples of that TSCBN to learn a TSCBN structure. By comparison of the given ground truth model and the learned TSCBN evaluation is performed.

Parameterization The temporal gap between subsequent state changes of a TV is randomly drawn between 0.5 and 1.0 from a uniform distribution. The state change probability is set to 0.95, the number of states per TV to 3, number of TVs to $n_{TV} = 5$ and length per TV to $n_L = 4$ and $0.5 \cdot n_L \cdot n_{TV}$ random inter-edges and 5000 MSSs sampled for learning. The structure learned by each algorithm was compared to the structure of the original network. TrieDiscover is parameterized with $k = 0.1$ to be able to filter the noisy parts, t_{th} to 1.0 as gaps between intra nodes are drawn from Gaussians with mean 0.5. In the sbTD BIC is used as score, in cbTD and cbvTD $\alpha = 0.01$ and $\chi_{thr} = 1.0$. In the results given in Fig. 14 we adjusted the three parameters above and used $n_{TV} = 3$, length $n_L = 3$ and $t_{th} = 0.5$.

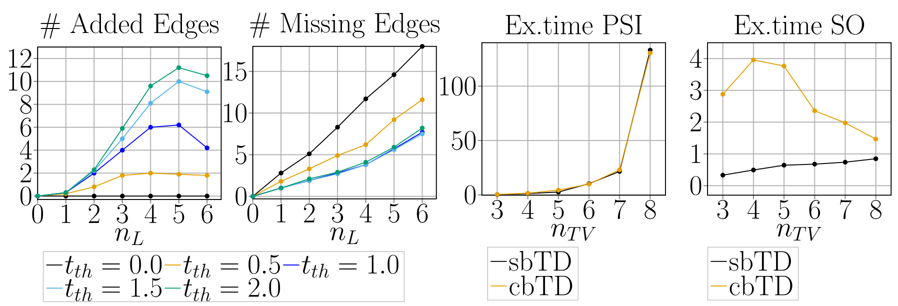


Fig. 14 Left: Various number of added and missing edges for sbTD in comparison to ground truth when structure and t_{th} is varied, with 3 TVs assumed. Mid and right: Execution time when assuming $n_L = 3$ for various numbers of TVs and the two steps of TrieDiscover

Evaluated approaches In total we compared six different algorithms. First, those are our approaches with different structure optimizations as proposed in Sect. 3, which are cbTD, cbvTD and sbTD. During Structure Optimization TrieDiscover uses an exact score optimization approach called *parent graph*, which was presented in Yuan and Malone (2013). It calculates the score for small parent sets first. If a parent set is non-optimal, also parent sets that include the non-optimal parent set are non-optimal. A heuristic is used to decide for which parent set the next score is calculated. Second, as baselines we use greedy hill climbing (GHC), the PC algorithm (PC) and the hybrid max-min hill climbing algorithm (MMHC). We chose static approaches here as no existing dynamical approach is directly applicable to generate TSCBNs, while with static methods correlations can be found when nodes per TV are provided. Further, to make those applicable the assignment of state changes to temporal nodes in the TSCBN has to be done prior to execution. We decided to use a naive assignment (ith event of signal X is assigned to node X_i) to evaluate if the advanced assignment of our algorithm leads to better results. No structural constraints are implanted in the initial networks as this may lead to cycles. The PC algorithm usually returns a partially directed structure. We decided to orient the edges using the index number of the sequential nodes per TV (from smaller index number to the larger one). If both index numbers are identical, an arbitrary orientation is chosen.
The significance level in the constraint-based and hybrid approaches was chosen from the values $\alpha = 0.01$, $\alpha = 0.05$, and $\alpha = 0.1$. Good results were achieved when using $\alpha = 0.01$ in the PC, MMHC, cbTD and cbvTD algorithms. Also, a maximal condition set size of two nodes turned out to be a good tradeoff between precision and performance when executing the cbTD and PC algorithm. All algorithms were implemented in Python, while GHC, PC, and MMHC were taken from the Python package pgmpy.[2]

Evaluation criteria As a computational performance metric we use the *runtime* of the algorithms.
To measure precision *Structural Hamming Distance (SHD)* is used (Tsamardinos et al. 2006), which is a common metric for comparing discovered to original network

[2] http://pgmpy.org/.

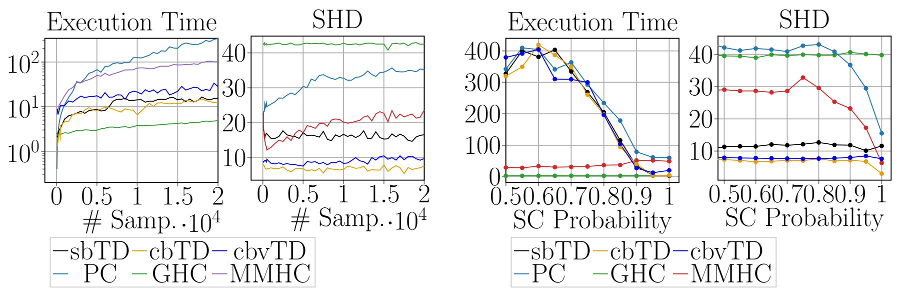


Fig. 15 Results in terms of runtime and SHD for various sizes of the training set and various SC probabilities

structures. The SHD is given by the sum of the number of missing edges, the number of additional edges and the number of wrongly oriented edges. Thus, a small SHD indicates a higher similarity between networks. Further, the number of edges that were wrongly added and missing edges are used.

Experiments In the first experiments in Fig. 14 we evaluated the proposed discovery approaches individually in terms of the influence of t_{th} and the structure size. Next, we compared our approach to the baselines when different numbers of samples are used for training and different SC probablities are given. The results of this experiment are shown in Fig. 15.

5.4.2 Results

Varying parameters of TrieDiscover Varying n_L and t_{th} gave the results shown in Fig. 14 (left two). It can be seen that small t_{th} lead to less edges, i.e. many missing edges, as the candidate set is empty. However, with increasing t_{th} the number of missing edges decreases. Too pessimistic t_{th} lead to an increase of added edges, as the structure optimization step yields a large number of edges that are spurious dependencies, e.g. given an edge $A_2 \rightarrow B_2$, strong dependencies exist between A_2 and preceding events. This may lead to spurious dependence between A_1 and B_2. Further, it can be seen that for increasing n_L the absolute number of bad edges increases, while when putting it into relation to the structure the relative increase is less. Thus, the choice of t_{th} is important to achieve good precision. In Sect. 6 we will show how this value can be found empirically by inspection of the input MSSs.
Next, we discuss the results when varying the sample size, the SC probability and n_{TV}.

Varying number of TVs When the number of TVs is increased the execution time grows as well, which is shown in Fig. 14 (two right). There parent set identification becomes increasingly complex, as the complexity of the trie network grows with more potential combinations. This requires more operations such as merging, and thus, more time. For instance for the case of 8 TVs it makes up around 95% of runtime. The structure optimization is less complex, when good subsets are found in the first step. Especially during parent set identification, e.g. with s TVs and m nodes per TV, this is as the number of potential paths in the trie is $s^{s \cdot m}$. Hence, increasing the number of

TVs has a worse impact on the number of paths in the trie than increasing the number of nodes. Thus an increase of n_{TV} and n_L make the minimization of the trie inefficient.

Varying sample size The size of the training set does not meaningfully improve precision of TrieDiscover as Fig. 15 (left) shows. This is as for the given network structure already less samples suffice to discover paths in the trie as we capture those losslessly. MMHC and PC, however improve with more samples as correlations become clearer with this.
In terms of runtime GHC performs best, while it is least precise. Our three proposed approaches are about 10 times faster than the constraint-based baselines and also good in precision even for less samples given.

Varying SC probability The SC probability sets the noise in the data during learning. Varying the SC probability gives the results shown in Fig. 15 (right two). By looking at the SHD it can be seen that all algorithms perform worse for smaller SC probabilities, i.e. if more latency is present. Overall all three variants of TrieDiscover yield the best results here. Especially if a higher degree of latency is present, our approach still yields good results. This is as our trie is built up exactly, i.e. even if paths are seen rarely, those are present as candidates for structure optimization.
Moreover, the naive event-to node assignment used in the baselines however, results in unclear relations between two events, e.g. for a sequence $\langle A, A, B, A\rangle$, the second A is latent thus, the third A will be assumed A_2. This in turn weakens the correlations computed between such state changes and makes it hard for static approaches to discover those. This effect becomes less meaningful with less latency. Thus, for the case of no latency the static approaches improve significantly performing similar to our approaches. Additionally, the discovered structure of the static baseline algorithms suffer from wrongly oriented edges, as we assumed a naive orientation rule for those algorithms. This is overcome in TrieDiscover by including control flow information.

Also, constraint-based approaches outperform score-based approaches in terms of additional edges as the latter tend to contain spurious dependencies. The further, use CI tests that are performed conditioned on a set of other nodes, which allows to find and remove edges with spurious dependencies. This is also why cbTD works better than sbTD.

In terms of runtime GHC and MMHC perform best, while all other approaches yield higher runtimes with more latency. This is as more latency also means more variation in possible paths represented in the data set. For the proposed approaches this results in more merging operations required. This could be improved in the future by using more approximative merging techniques.

Conclusion The proposed approach allows to successfully discover a meaningful structure of TSCBNs. In comparison to the baseline approaches it is most precise in discovering structures. Further, it yields higher runtimes if more latency is present, while it is similar to the baseline approaches if only fully observed MSSs are given.

6 Case study: automotive fault diagnosis

In this section we demonstrate how TSCBNs can be used for the effective extraction of specifications from MSSs. For this we analyzed a real world data set recorded from a fleet of test vehicles of a big automobile company. This is done with TrieDiscover and EM to learn the TSCBN followed by BaySpec (Mrowca et al. 2019) and MPE for Specification Extraction.

6.1 Background

In modern vehicles communication between Electronic Control Units (ECUs) is performed within its intra-vehicular networks to execute functionality, e.g. sensor data is exchanged during wiper usage (Mrowca et al. 2018a). Analyzing such data in terms of correctness is essential to guarantee functional quality of subsystems at a low cost. Such traces of vehicles are recorded as signals, which are sent between ECUs. Each signal is one piece of information, i.e. events, states or state changes of the car, such as the acceleration pedal, the brake state or the GPS position. Those signals can be interdependent. Modeling causal dependency between such signals allows to capture the vehicle's state and to infer specifications.
Signals are usually sent cyclically within milliseconds and potentially with identical signal values. Including all of those signals is intractable due to a high number of signals. This requires to remove duplicate subsequent signals, ending up with state changes per signal only. Thus, the communication behavior can be seen as a MSS with each signal as TV. An important task to ensure functional correctness of subsystems is the identification of relevant specifications. Finding those however, is challenging as functional complexity increases and thus, manual definition of specification becomes intractable. Hence, automated specification mining approaches are required to solve this. TSCBNs can be well used for this task. That is, given MSSs of observed behavior of a functional process, TSCBNs can be learned. We demonstrate this in the following using two functions.

6.2 Data sets

The given data set represents the functionality of the indicator lights in a car. The indicator light is switched *on* by a driver using the handle bar, when turning left or right. This function consists of a handle S_{bar} that *(de-)activates* the indicator either in *steady mode* or for *3 seconds*, indicated in S_{type}. This influences the state of the indicator S_{state}, e.g. *right indicator on*. Depending on this, in steady mode synchronization S_{sync} is *started* and a *new indication cycle* S_{cycle} begins. Once S_{bar} changes back to its *default state* from the *3 second* state, depending on this state, synchronization is *stopped*. If it is in *steady mode* it is only *stopped* later at S_{bar}, when being returned to *default* by the user.
Here, 28.6% of the observed sequences remain in an identical state over multiple intervals, which shows that our latent parameter estimation is required. Properties of the data set are given in Table 1.

Table 1 Properties of the automotive data set including results after model creation and parameter estimation

Data set	# Samples	# Temp. variables	# States/# nodes	Latent sequences	Mean log-likelihood	Temporal log-likelihood
Indicator	104	5	3.67	28.6%	− 0.23	-44.78

6.3 Fully automated specification extraction

6.3.1 Setup

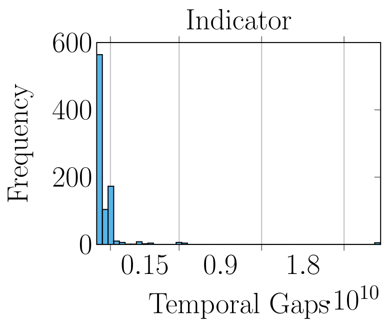
Here, we demonstrate how TSCBNs can be used for automated specification mining to infer the following two types of specifications. First, those are LTL specifications deduced from combinations of paths within a TSCBN which capture the most likely sequential behavior between paths along TVs of the learned process. Second, those are most dominant system states which resemble the most likely overall process progression when all TVs are taken into account at the same time.

Experiment We train the TSCBN from the data set using TrieDiscover for structure discovery and the EM approach for parameter estimation. On the resulting TSCBN we once, apply the BaySpec (Mrowca et al. 2019) algorithm to deduce LTL specifications and as an alternative find the Most Probable Explanation (MPE) of the network to deduce dominant process states as potential specifications.

Most probable explanation The MPE is an estimate of the unknown latent dominant system state, which the modeled process represents. We assume that a MPE estimate in our case resembles the most likely constellation of the process. This can either be interpreted as a potential specification of the system or, during fault diagnosis, as the prevalent system state that might have lead to a fault. For the latter, segments that contain a known fault could have been taken for training and the cause of the error could be found with MPE. We compute the n most likely MPE estimates in state and time by sampling 100 000 times from the trained TSCBN and by counting the occurrence frequencies of drawn state (excl. temporal nodes) combinations. Most likely drawn combinations are then, ranked according to its occurrence frequencies. Each state combination outcome is specified in time by setting the combinations as evidence in the discrete state nodes, and reading the corresponding learned μ values, under given evidence, from the temporal nodes. This gives a specification in state and time with associated likelihoods per outcome combination.

BaySpec BaySpec is a Specification Mining algorithm, that allows to learn LTL specifications from BNs. This is done by first, looking for most probable paths and then, subsequently merging those paths until specifications of appropriate strictness are found. We used the implementation provided by the authors of Mrowca et al. (2019).[3] BaySpec finds formulas of the type *premise* $\mapsto$ *conclusion*, where the premise is a unique combination within the trace, that if present must have the conclusion follow. If this is not the case the specification is violated.

[3] https://github.com/arturmrowca/bayspec.



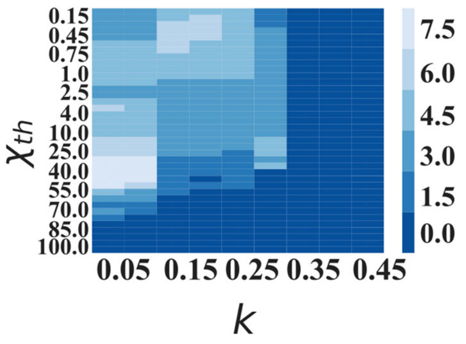


Fig. 16 Left: Distribution of gaps between consecutive state changes in nanoseconds. Right: Number of inter-edges after structure discovery for different values of k and χ_{th}

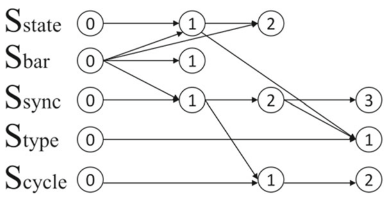


Fig. 17 This figure shows the network that was discovered with TrieDiscover

Parameterization We used cbvTD which has parameters t_{th}, k and χ_{th}. t_{th} of TrieDiscover is found by inspecting the distribution of gaps between consecutive events in the data set as shown in Fig. 16. To avoid ignorance of potentially relevant temporal dependencies we chose t_{th} pessimistically to include the last two preceding events (i.e. twice the highest gap size), i.e. for indicator $t_{th} = 2 \cdot 4 \cdot 10^9$. Then, we ran the approach for values of k between 0.05 and 1.00 and for χ_{th} between 0.05 and 100.00 to find k and χ_{th} that yield a medium number of inter-edges, i.e. as k removes noisy events and χ_{th} defines the required strength of correlations between events both values have a high impact on the quality of the algorithm's performance on real world data. Results of this are shown in Fig. 16. For the indicator we chose $k = 0.2$ and $\chi_{th} = 0.25$. We ran the EM Algorithm with 5 iterations for parameter estimation. For BaySpec we used the metric-based approach with a minimum likelihood threshold of 0.6. Lastly, certain state changes occur at exactly the same time, while directions of dependencies matter. This is because such information is recorded at similar sampling rates, e.g. to an expert it is obvious that the detection of opposing traffic causes to change the control of the high beam state. Therefore, we define with an expert the order between some TVs if a correlation is found. The discovered network is shown in Fig. 17, excerpts of the most likely discovered LTL specifications are shown in Table 2, its metrics are provided in Table 3 and excerpts of the most likely MPE estimates are provided in the results section.

Metrics The found LTL specifications are evaluated in terms of complexity as used in Mrowca et al. (2019), in terms of height and width of the specifications. Height resembles the depth of the formula (i.e. deepest number of consequent nested operators), while the width is the number of unique literals in the formula. Further, an

Table 2 Excerpt of LTL specifications found with BaySpec for the indicator data set with likelihoods of found specifications $\mathcal{L}$

$\mathcal{L}$ (%)	Discovered LTL—specifications
87.37	$\mathbf{G}((S_{bar}$: tip up $\vee$ S_{sync}:cyc. continues) $\mapsto$ $\mathbf{XG}((S_{sync}$:new cyc. $\mapsto$ $\mathbf{X}((S_{sync}$:sync $\wedge$ $\mathbf{X}(S_{sync}$:cyc. continues))))))
86.90	$\mathbf{G}((S_{bar}$:overtip down $\vee$ S_{bar}:tip up) $\mapsto$ $\mathbf{X}(\mathbf{F}(S_{bar}$:no action)))
86.36	$\mathbf{G}((S_{bar}$:tip down $\vee$ S_{bar}:tip up) $\mapsto$ $\mathbf{X}(\mathbf{F}((S_{sync}$:new cyc. $\wedge$ $\mathbf{X}(S_{sync}$:sync $\wedge$ X$(S_{sync}$:cyc. continues))))))

expert manually labeled found specifications as either fully right or only partly right. The latter implies that at least one literal is placed badly, such that malicious firing of the specification might occur. The accuracy resembles the ratio of specifications that are fully right. Further, from the accurate traces we labeled the found specifications as either trivial, i.e. specifications that always fire, or relevant, i.e. specifications that fire in a critical case which might reveal an anomalous spot in the trace.

6.3.2 Results

We evaluate the learned model structure, the found LTL specifications and the found MPE estimates. As shown in Table 1, good results, with low likelihoods in state and time are found when modeling the given data with TSCBNs. This shows that the data set can be modeled expressively with TSCBNs. Please also note that overfitting is a desired property in our scenario, as we do not aim for generalization but rather for model fitness.

Model structure The discovered model is shown in Fig. 17. Overall the discovered *indicator* structure seems intuitive and plausible. The edges rooting in S_{bar} influence S_{state} and S_{sync}, which is as expected, as the handle bar's initial value - i.e. the type of action - determines the state the indicator will change to and if a synchronization is initialized. Further, S_{sync} continues to influence S_{cycle} based on its evidence, which is also true as an activation of synchronization starts a new cycle or continues it depending on S_{cycle} preceding state. Also, dependent on the synchronization type S_{sync} it can be read off S_{type} if a permanent or a non-permanent indication is present. The edge between S_{bar} 0 and S_{state} 2 seems surprising at first. However, it makes sense as a normal pushing of the handle bar causes the indicator to turn on at 1 and return to its default state after some time, which is within the training sequence, while if a further push is applied on the bar the indicator will permanently blink, i.e. state 2 will remain in on state.

BaySpec LTL specifications As Table 3 shows BaySpec found 28 specifications from the indicator TSCBN. Some examples of specifications are given in 2, while metrics found over all specifications are provided in Table 3, e.g. the first LTL formula suggests that if the handle bar was tipped up and the synchronization started a new cycle, the synchronization has to continue, before then, a new cycle is started. The second formula implies that a movement of the handle bar needs to return to its default state, while the last formula says that the synchronization sequence of the first formula is started once

Table 3 Metrics for the specifications that were extracted with BaySpec

Data set	# LTL spec.	$\varnothing height$	$\varnothing width$	Accuracy (%)	Relevant (%)
Indicator	28	5.75	4.03	67.85	84.22

the handle bar was triggered by the user. In Table 3 we see that a good accuracy is reached. The discrepancy to 100% can be explained with an imbalance in the data set. That is, in the indicator data sets users tend to use the short indication far more often than the permanent indication. Therefore, the non permanent case is well represented while the permanent case yields CPDs that are mostly approximated during the EM approach. This, can be adjusted in future applications by either including balancing or including expert input at this stage. The complexity and expressiveness of the found specifications yields a good quality, with a high percentage of relevant specifications and specifications of heights around 5 and 3 to 4 symbols included per specification. This shows, that TSCBNs can be used to automatically learn LTL specifications from observed system behavior. This allows to lower manual effort during specification generation.

MPE estimations—indicator For the indicator frequencies are within 11.99% and 0.017%. The two most likely estimates are with 11.99% frequency

$S_{bar} = \langle(tip\ up, \Delta t = 0.0s)\ , (no\ action, \Delta t = 0.513s)\rangle$
$S_{sync} = \langle(cyc.\ cont., \Delta t = 0.0s)\ , (new\ cyc., \Delta t = 0.017s), (sync, \Delta t = 0.639s), (cyc.\ cont., \Delta t = 2.052s)\rangle$
$S_{type} = \langle(non\ permanent, \Delta t = 0.0s), (non\ permanent, \Delta t = 0.0s)\rangle$
$S_{state} = \langle(both\ off, \Delta t = 0.0s), (right\ on, \Delta t = 0.017s), (both\ off, \Delta t = 1.682s)\rangle$
$S_{cycle} = \langle(no\ ind., \Delta t = 0.0s), (normal, \Delta t = 0.0s), (no\ ind., \Delta t = 2.660s)\rangle$

and with 4.4% frequency

$S_{bar} = \langle(tip\ down, \Delta t = 0.0), (no\ action, \Delta t = 0.572s)\rangle$
$S_{sync} = \langle(cyc.\ cont., \Delta t = 0.0), (new\ cyc., \Delta t = 0.020s), (sync, \Delta t = 0.639s), (cyc.\ cont., \Delta t = 2.052s)\rangle$
$S_{type} = \langle(non\ permanent, \Delta t = 0.0s), (non\ permanent, \Delta t = 0.0s)\rangle$
$S_{state} = \langle(both\ off, \Delta t = 0.0s), (left\ on, \Delta t = 0.0204s), (both\ off, \Delta t = 1.675s)\rangle$
$S_{cycle} = \langle(no\ ind., \Delta t = 0.0s), (normal, \Delta t = 0.181s), (no\ ind., \Delta t = 2.660s)\rangle$

For this case the indicator learned both the activation of the right indicator after the handle bar was *tipped up* and the left activation when *tipping down*. Both caused a *new indication cycle* to start. Notably, here the timing was learned nearly exact, e.g. it is plausible that after *tipping up* the indicator starts 0.017 seconds later and *stops* indication after around 1.7 seconds (S_{state} *2*) and confirms this at around 2.6 seconds in S_{cycle} *2* with the *no indication* signal. In practice, this approach allows an expert to first, directly understand the prevalent system behaviors (e.g. for fault diagnosis) and to second, deduce potential specifications in state and time in an automated manner.

This is, as snapshots of process variants of MSSs can be compactly represented as outcomes of learned TSCBNs.
In the future this opens up further applications, such as in prediction tasks, e.g. when setting observed initial states of the multidimensional indicator state as evidence in the network and reading of Maximum Aposteriori Hypothesis (MAP) estimates or likely paths, as in BaySpec, from the network.

7 Conclusion

We introduced TSCBNs as an effective way to model MSSs, as well as structure and parameter learning approaches for this model. In an extensive evaluation we showed that the proposed approaches improve the state of the art.

Future work With the introduced model several applications for temporal diagnosis and prediction, such as anomaly detection or deduction of test cases, become available. However, for this specialized inference algorithms need to be developed, e.g. as TSCBNs represent observed processes, setting evidence of subparts of it and inspection of unlikely outcomes resulting from it allows to identify anomalies and to draw conclusions for diagnosis. Also, unlikely paths could be searched for. In the same way, given evidence conclusions about most likely successive outcomes can be used for prediction. Future work includes approaches that allow to infer such anomalies from TSCBNs and to predict processes in state and time. Further directions include the extraction of multidimensional specifications and the application of those on TSCBNs for verification. Mrowca et al. (2019) one dimensional specifications are extracted by transforming TSCBNs into a mining graph, where most probable paths can be deduced as specifications. Lastly, advanced statistics could be extracted from TSCBNs. Often the usage of a system guides the design of the product, i.e. in which states and under which conditions and in which order a system is used. This can also be modeled with TSCBNs and allows to deduce temporal statistics, e.g. the most probable orders of usage.
Further, we introduced EM, VI and MLE-R as parameter estimation approaches. Those can be further improved in terms of scalability by developing efficient implementations for GPU based computations.
TrieDiscover grows in complexity and thus, in runtime when the number of TVs is increased. This is mainly due to the effort of minimization of the trie. Minimizing the trie is done to enable a lossless compression of the found sequences. Thus, future work might replace this step with a lossy compression of the event sequences which we expect to improve the run time of TrieDiscover significantly. Lastly, most modeling approaches focus on either modeling state sequences, continuous data (Hallac et al. 2017; Nodelman et al. 2002) or discrete events (Galán et al. 2002), while it would be interesting to model interdependence between all three temporal types combined in order to capture dependence of states on evolving events and time-series. Although, the proposed approaches already provide good scalability for models of up to about

50 nodes, this needs still to be improved. This can e.g. be achieved by using lossy SD approaches or by adding more constraints to the BN.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix A

Preliminaries: Bayesian networks

BNs are DAGs that represent causal relationships between RVs X_i as nodes and their dependencies as edges. Such networks define a Joint Probability Distribution (JPD) $P(X_1, X_2, ..., X_n)$ over all n RVs in the graph, which allows for various types of probabilistic inference tasks, including diagnosis or prediction. JPDs are computed from Conditional Probability Distributions (CPD) $P(X_i|Pa(X_i))$ which are defined per RV X_i in the network conditioned on its respective parents $Pa(X_i)$. For a set of variables $X_1, X_2, ..., X_n$ the JPD is computed with

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)). \tag{20}$$

To infer probability distributions along the network under given evidence approaches such as marginalization of JPDs or Bayes' theorem can be used.
Mostly, for ease of computation of JPDs, RVs are assumed to be either all of discrete or all of continuous type. If both types occur in the network simultaneously, the graph is called a HBN. In HBNs the set of RVs X is divided into two sets $X = Y \cup Z$, where Y represents discrete and Z continuous RVs. In this work we restrict our notion of HBNs to the case, where continuous RVs can only be conditioned on discrete ones and the inverted case is not allowed. A common way for inference in such cases is to assume fixed distributions (e.g. *Gaussian* or *Exponential*) defined with parameters Θ per continuous node Z and to condition each Θ on Z's discrete parents, e.g. assuming one discrete parent Y, each parent outcome y gives a distribution Σ for the respective node. That is,

$$Z|Y = y \mapsto \Sigma(\Theta(y)). \tag{21}$$

In this work the notion of HBNs is extended to a general temporal model, that allows to compactly model intervals and its dependencies in MSSs.

Discussion: comparison of TSCBNs to existing approaches

In this section extended details are provided for the discussion of 2.5. This is also done on the basis of the scenario in Fig. 3.

Compactness and parameter complexity Compact representations are desireable, as a more compact model requires to train less parameters and allows for better interpretability. Process models are complex in size, as those model each state change with each outcome as a separate node, e.g. in Fig. 3 for S_{dist} at both node 0 and 1 there would be three nodes: $too\ near$, $too\ far$, ok. Time-sliced BNs repeat a static structure, that has edges between TVs anywhere where TSCBNs have those, i.e. not only at points where true causality is given. This fact and redundancy due to repetition make those complex, as well. Among interval-based approaches TBNEs (Arroyo-Figueroa et al. 2005) are comparably compact, while containing less temporal information. Other approaches are either of qualitative nature (e.g. MTBNs Aliferis et al. 1996) or do model variants of possible occurrences as a tree that explodes with the length of the process to model (Galán et al. 2002). CTBNs do require the same static structure as DBNs and to store all variants of connection combinations. This results in many condition combinations resulting in a high parameter space.
In contrast to this, TSCBNs model state changs only at times that a change is possible. This for the first time enables multidimensional path analyses along actually possible paths (e.g. given evidence of a subprocess, to infer likely or unlikely behavior). Also, this representation allows for best interpretability, as temporal patterns can be directly read of, e.g. in Fig. 3 one can directly read that S_{dist} causes changes in S_{ctrl}, that in turn influences both the acceleration S_{acc} and braking S_{brake} actions, no matter what the outcomes of those nodes are. No other process model is able to provide this degree of expressiveness for MSSs.

Explicit modeling Existing approaches can model MSSs' processes either explicitly (e.g. Process Models) or implicitly (e.g. tree-like BNs Galán et al. 2000). The further, is advantageous when analyzing behavioral patterns in data, while the latter is mostly suited to answer specific types of questions (e.g. diagnose a disease given symptoms in temporal order). Existing explicit models include TBNEs, Process Models or unrolled DBNs. TBNEs build up a tree like structure to capture the process flow and do not model time as continuous nodes making it less expressive. Process Models do have high numbers of edges, while the repetition in DBNs is not able to explicitly represent the actual flow of the process. TSCBNs are the first BNs that provide an expressive unrolled representation of the whole modeled process in multiple dimensions in state and time. This enables effective multidimensional inference. In particular, the shape of TSCBNs represents a process of multiple MSSs as a JPD of the overall process in state and time. This enables new types of analyses, such as finding rare state changes as anomalies given any set of assumed evidence.

Multidimensional procedural inference By representing the multidimensional nature of the data, effective evidence based inference is possible, as it was described before. For example in Fig. 3 the aim could be to compare the TSCBN distributions in $Sport$ mode against the ones in ECO mode. In TSCBNs this can be achieved by first computing most likely constellations given each respective evidence setting, yielding one MPE estimate for *Sport* and a MPE estimate for *ECO* mode. By comparing the same state nodes in those two cases differences can be read off, i.e. looking at node 1 of S_{acc} for $S_{mode} = Sport$ this is $high$, while for ECO it is low. In process models this would require to compare and search multiple paths, in DBNs all repetitions would need to be compared and in CTBNs multiple samples would need to be generated and compared per case. The same holds for the temporal behavior. Here, in TSCBNs relative timings can be compared with the same approach. For DBNs it would be required to inspect multiple repeated structures to check at which a change happens. For CTBNs sampling would again be needed and in Process Models multiple paths need to be analyzed.

Dynamic dependencies TSCBNs are the first explicit multidimensional BNs with an explicit process representation to assume that causality between states does change in time. Process models assume this, too, while those suffer from its one dimensional nature in this scenario. Comparable models, such as DBNs and CTBNs, assume static structures, that either result in an overhead of edges (e.g. in DBNs) or reduced variability in temporal modeling (e.g. in CTBNs). The assumption of dynamic dependence is a matter of definition, as in general TVs that are causally dependent keep this property permanently, while in our simplified assumption TSCBNs only keep causal dependencies at times that those do influence another state. This yields no loss of information, while resulting in better expressiveness. As stated before, this allows for more efficient inference, as only relevant paths of causality are analyzed in TSCBNs. In particular the static nature of CTBNs requires to store matrices per condition. However, if causality changes for a defined condition, the matrix cannot have two different values, making it impossible to model longer processes with dynamic dependencies.

Expressive interval modeling As laid out above, in real world data, per TV latent state changes can occur, e.g. in Fig. 3 S_{acc} remained in the *off* state, while it is causally dependent on S_{ctrl} at this stage. Thus, under different conditions a state change of S_{acc} to $high$ would have been possible. TSCBNs are the first process models to represent such latent dependencies. With this, intervals and causality between intervals (even if those do not change its state) is represented effectively. In DBNs this is represented through repetition of structure and in CTBNs it is implicitly represented through conditions of previous states. However, unlike in TSCBNs for those models it is not possible to perform explicit inference on this latent node, e.g. in Fig. 3 one might perform evidence based inference to compare behavior when S_{acc} remained off against when it remained low. TBNEs do not include such interval relations, too, as state changes that were not observed cannot be included in the model. Thus, this latent influence is not represented. With this, TSCBNs are the first models to allow to perform explicit evidence based inference of processes on intervals in state and time under uncertainty.

Continuous temporal modeling Representing time continuously allows for better representation of temporal behavior. In TSCBNs timings can be directly analyzed by setting evidence on the TSCBN and by inspecting the distributions at each node, e.g. one might set $S_{ctrl} = speed\ down$ at node 1 and $S_{brake} = inactive$ at node 0. Then, the μ of the defined Gaussian directly tells the average time the brake takes to change to $S_{brake} = active$ in node 1 after the control signal occurred. This, is less precise in DBNs and CTBNs, as timings are represented through discretization and statically conditioned matrices respectively, e.g. for DBNs it would be required to determine the most likely static repetition at which the control signal occurred and at which the brake changed. A similar approach is required in CTBNs. Process models are similarly able to determine such timings. However, here the respective paths would need to be analyzed, which is far more complex, then reading it from the structure as it is the case in TSCBNs.

Appendix B

TrieDiscover: complexity analysis

Let m be the number of sequences in $\hat{M}$ and L be the average length of the sequences. With this, the complexity for each step can be given as:

1. **Trie creation** During creation for every sequence in average L nodes are created yielding complexity $\mathcal{O}(m \cdot L)$. During filtering, the approach needs to check if an outgoing edge is removed for every node (i.e. at worst $m \cdot L$ nodes) resulting in a complexity of $O(m \cdot L \cdot d)$, where d is the maximum output degree. This degree is usually restricted yielding constant d and thus, complexity $O(m \cdot L)$.
2. **Minimization through subtree merging** This step has linear runtime w.r.t. the number of nodes which is at worst $\mathcal{O}(m \cdot L)$.
3. **Node indexing** Topological sorting has complexity $\mathcal{O}(numb_{nodes} + numb_{edges})$. At worst we have $m \cdot L$ nodes and $m \cdot (L - 1)$ edges, yielding $\mathcal{O}(m \cdot L)$. During the provisional index assignment we use topological sorting, and for each node an indexing and passing the set to the next nodes for all outgoing edges, giving $\mathcal{O}(m \cdot L \cdot d)$ or $\mathcal{O}(m \cdot L)$ if d is assumed constant. During refinement there is one traversal $\mathcal{O}(m \cdot L)$. Further, we assume h number of refinements required, where the SCC approach we use has $\mathcal{O}(numb_{nodes} + numb_{edges})$, which in our case is again $\mathcal{O}(m \cdot L)$, i.e. for h refinements $\mathcal{O}(m \cdot L \cdot h)$. Assuming a constant number of refinements this yields $\mathcal{O}(m \cdot L)$. Thus, for both steps combined we get an overall complexity of $\mathcal{O}(2 \cdot m \cdot L) = \mathcal{O}(m \cdot L)$.
4. **Parent candidate identification** For every ambiguous assignment in the set union step, a SCC algorithm has to be run, which gives for constant h of those assignments $\mathcal{O}(h \cdot m \cdot L) = O(m \cdot L)$. The temporal filtering is done per node resulting in complexity $\mathcal{O}(m \cdot L)$.

Overall complexity The overall complexity of all steps, except for structure optimization, is $\mathcal{O}(m \cdot L)$. For the optimization step existing methods are used, which determine the additional complexity. The improvement of TD depends on the number of potential edges that are identified. At worst every node could have the respective maximally possible number of parents in its parent set, yielding no benefit of TD. On the contrary, at best, the optimal set of parents is found per node. Using for instance score-based structure optimization, this would result in complexity $\mathcal{O}(m \cdot L \cdot p)$, where p is the average parent set size. Yielding $\mathcal{O}(m \cdot L)$ for a limited number p of parents per node.

TrieDiscover: pseudocode

Let PODF($\mathcal{T}$) be the post-order depth-first traversal of $\mathcal{T}$. Let Top($\mathcal{T}$) be a topological sorting of $\mathcal{T}$. Let RevTop($\mathcal{T}$) be a reversed topological sorting of $\mathcal{T}$. Let Cand(X) be the set of parent candidates of a node X. With this, the pseudo code of Algorithm 1 is given.

Algorithm 1 TrieDiscover

```
1: for every sequence $\hat{M}_i, i \in [1, m]$ in $\mathcal{M}_{obs}$ do          ▷ Trie Creation
2:     for every TV $S_i, i \in [1, l]$ in $\hat{M}_i$ do
3:         Add $S_i$ to trie $\mathcal{T}$ if required.
4: for every node $X$ in $\mathcal{T}$ do
5:     for every outgoing edge $e$ of $X$ do
6:         Remove edge $e$ from $\mathcal{T}$ if $frequency(e) < k \cdot .frequency(X)$
7: for every node $X$ in PODF($\mathcal{T}$) do          ▷ Subtree Merging
8:     Assign sub-tree code to $X$.
9:     Hash $X$ based on the sub-tree code.
10:    Merge $X$ with other node in case of hash collision.
11: Initialize parents set of root of $\mathcal{T}$ to all initial nodes.          ▷ Node Indexing
12: for every node $X$ in Top($\mathcal{T}$) do
13:    Merge parents sets passed by predecessors.
14:    Add node to set based on the latest state change of $X$'s TV.
15:    Pass updated parents set to all successors.
16: Initialize successors set of leaves of $\mathcal{T}$ to empty set.
17: for every node $X$ in RevTop($\mathcal{T}$) do
18:    Merge successors sets passed by successors.
19:    Identify ambiguous assignments and resolve based on SCCs.
20:    Pass updated successors set to all predecessors.
21: Merge nodes of the same TV and occurrence.          ▷ Parent Candidate Identification
22: for every node $X$ in $\mathcal{T}$ do
23:    Update Cand$(X)$ to the union of parents sets of the merged nodes.
24: for every SCC in $\mathcal{T}$ do
25:    Remove nodes in the SCC from each other's Cand's.
26: for every node $X$ in $\mathcal{T}$ do
27:    Remove nodes from Cand$(X)$ based on the temporal threshold $t_{th}$.
```

Appendix C

Non-latent estimation

Non-latent Bayesian estimation and MLE

If there are no latent state changes, classical parameter estimation can be used. This is the case, when we assume that all states changed in a MSS or when we aim to model fully observed multivariate event sequences. Such parameter estimation can be MLE or Bayesian based. We further estimate states and temporal nodes separately as temporal nodes are leaf nodes.

State nodes can be estimated in the following manner.

State nodes—MLE The goal of MLE is to set the parameters Θ of the TSCBN such that those maximize the likelihood of the data. These ML estimates Θ^* of the state structure of a TSCBN can be computed independently under the i.i.d. assumption. For this, according to Barber (2012) it is possible to decompose the network structure, such that per node v_{ij} and parent outcome set $Pa(v_{ij}) = t$ the following holds for each CPD

$$p(v_{ij} = s|Pa(v_{ij}) = t) \propto \sum_{n=0}^{N} \mathbb{I}[v_{ij}^n = s|Pa(v_{ij}^n) = t]. \tag{22}$$

This corresponds to the fact, that the CPD entry $p(v_{ij} = s|Pa(v_{ij}) = t)$ can be set by counting the number of times $\{v_{ij} = s|Pa(v_{ij}) = t]\}$ was seen in the data set Z for fixed joint parental state t (Barber 2012).
For bigger networks data fragmentation becomes problematic as for higher numbers of parameters more samples need to be observed. This also means that few samples for parameter estimation tend to overfit the data (Karkera 2014). Fragmentation decreases with less parent nodes and discrete parent values. TSCBNs are designed to minimize the further, while the latter needs to be restricted.

State nodes—Bayesian approach According to Barber (2012) in this approach each node is governed by a parameter which is assumed to have a distribution itself, with the JPD of the network as $p(z_1, z_2, ...)$. This can be represented with its parameters as

$$p(z_1, z_2, ...) = p(z_1|Pa^*(z_1); \Theta_{z1})p(z_2|Pa^*(z_2); \Theta_{z2})... \tag{23}$$
$$= \Theta_{z1}^{Pa^*(z_1)}\Theta_{z2}^{Pa^*(z_2)}... \tag{24}$$

where z_i are all nodes v_{ij} of the TSCBN and $Pa^*(z_i)$ is a defined conditional set of z_i's parents. Assuming global parameter independence and observations $\mathbf{X}$ the posterior can be represented as

$$p(\Theta_{z1}, \Theta_{z2}, ...|\mathbf{X}) \propto p(\Theta_{z1}|x_1^*)p(\Theta_{z2}|x_2^*)... \tag{25}$$

where Θ_{zi} is the product of all CPDs of its node $\Theta_{zi}^{Pa^*(z_i)}$ and x_i^* the observation at node i. This allows to learn parameter posteriors separately. As each parameter Θ_{zi} is still multidimensional it is common to further assume local parameter independence, which is that Θ_{zi} is the product of all CPDs of its node $\Theta_{zi}^{Pa^*(z_i)}$. With this the posterior both factorizes over parental states and the local parameters, which means that

$$p(\Theta_{zi}|x_i^*) \propto p(x_i^*|\Theta_{zi})p(\Theta_{zi}^{Pa^*(z_i)_1})p(\Theta_{zi}^{Pa^*(z_i)_2})... \tag{26}$$

By assuming i.i.d. data and the local and global parameter prior independencies, Dirichlet priors can be used to now determine the parameters. This is explained in more detail in according literature (Barber 2012).

Temporal nodes Temporal nodes can be simply estimated by fitting Gaussian distributions over each node under the respective conditions.

Latent estimation

Challenges of latent estimation

Challenges The following challenges arise.
Consistency: Learning parameters of TSCBNs from partially observed MSSs is challenging, as subsets of nodes (i.e. TV sequences) are potentially interdependent, temporal consistency needs to be guaranteed and a valid combinatorical mapping from observations to latent estimates is required.
Complexity: Additionally, for the case of many parents complexity in TSCBNs grows, which requires scalability of approaches.
Ambiguity of Nodes: In a MSS an interval is defined by its state changes and so is the TSCBN. However, if at an expected state change of a TV the sequence remains in its previous state, this state change is not observed. Instead, the observed interval simply continues and the next, actual state change is observed. Thus, as shown in Fig. 4 the structure of a TSCBN may require more nodes per TV than were observed. In this case the problem occurs that assigning k observed state changes to n nodes is ambiguous, e.g. for $n = 5$ and an observed sequence $\langle A, B, A, C\rangle$ the actual state changes could be $\langle A, A, B, A, C\rangle$, $\langle A, B, B, A, C\rangle$, $\langle A, B, A, A, C\rangle$ or $\langle A, B, A, C, C\rangle$. In general $\binom{n-1}{k-1}$ combinations are possible.

Deduction of variational inference equations

A common approximation approach for latent variable models, such as TSCBNs is Variational Inference, which was introduced in the main part. Here, a detailed deduction of formulas is provided.
Similar to the EM algorithm, VI tries to estimate the posterior latent distribution and the posterior observed distribution. In VI this is achieved by approximating the global latent posterior distribution $p(Z|\Theta)$ with tractable local distribution estimates $q(Z)$. This is done through maximization of the Evidence Lower Bound (ELBO). Further, using the Mean field assumption we consider the local estimates as conditionally independent, which allows to compute individual latent variables separately. Also, unlike in EM, where we considered each latent sequence, in VI we consider each latent node v_{ij} and Δt_{ij}.

Variational inference in TSCBNs

Structure and mapping of latency For the VI approach, we assume the probabilistic structure shown in Fig. 4, but with inverted edges reaching from v_{ij} to X_i for all TVs. From this representation, we will deduce update equations to find estimates for all RVs v_{ij} and Δt_{ij}.
Notably, in this representation we are mapping observations to latent state node values. An important assumption that we introduce here is, that each observation X_o only allows for a subset of valid latent outcomes, which are represented in each v_{ij} of each X_i. Thus, per observation $X_o = (x_{o1}, x_{o2}, ...)$ only a subset of mappings from v_{ij} to

X_i are possible, e.g. for an observed $X_i = \langle A, B\rangle$ Z_i might only be ($v_{i1} = A, v_{i2} = A, v_{i3} = B$) or ($v_{i1} = A, v_{i2} = B, v_{i3} = B$). With this we define all such valid mappings from an observed $X_i = X_o$ to its latent Z_i as

$$p(X_i = X_o|Z_i) = \begin{cases} 1 \; if \;\; Z_i \mapsto X_o \;\; valid \\ 0 \; else \end{cases} \tag{27}$$

For the example above $p(X_i = \langle A, B\rangle|v_{i1} = A, v_{i2} = A, v_{i3} = B) = 1$, while for any bad mapping such as $Z_i = \langle A, B, C\rangle$ $p(X_i = \langle A, B\rangle|v_{i1} = A, v_{i2} = B, v_{i3} = C) = 0$ holds. Notably, when taking all dimensions (i.e. $X_1, X_2, ...$) into account, with this invalid MSSs combinations (that are formed by all Z_i combined) have a likelihood of 0 in the TSCBN.

ELBO derivation In VI the goal is to find the variational distribution $q(Z)$ that approximates the true posterior $p(Z|X)$ of the TSCBN.
According to Bishop et al. (2003) the likelihood of a general latent BN can be written as

$$p(Z, X) = \prod_n p(w_n|Pa(w_n)) \tag{28}$$

with w_n being all n nodes in the network, i.e. in TSCBNs either state nodes v or temporal nodes Δt.
Further, the probability $P(X)$ of any observation can be written in terms of the ELBO $\mathcal{L}$ and the KL divergence between the real and variational distribution as

$$ln(P(X)) = \mathcal{L}(q) + \mathbb{KL}(q||p) \tag{29}$$

$$\mathcal{L}(q) = \sum_Z q(Z|X) \cdot ln(\frac{p(Z, X)}{q(Z|X)}) \tag{30}$$

$$\mathbb{KL}(q||p) = -\sum_Z q(Z|X) ln(\frac{p(Z|X)}{q(Z|X)}) \tag{31}$$

where $\sum_Z$ is the sum over all network outcomes of Z (Bishop et al. 2003). With this, minimization of $\mathbb{KL}$ is equal to maximizing the ELBO which can be written as expectation

$$\mathcal{L}(q) = \mathbb{E}_{q(Z)}[\log p(X, Z) - \log q(Z)] \tag{32}$$

Mean field approximation We factorize the variational JPD $q(Z)$ to allow for local computations around nodes as

$$q(Z) = \prod_{i=0}^{M} \prod_{j=0}^{M_i} q(z_{ij}) \tag{33}$$

where z_{ij} can be a discrete state node v_{ij} or a continuous temporal node Δt_{ij}.

With this, maximization of the overall ELBO (i.e. finding good approximations of parameters per node) is similar to maximization of the ELBO of the factorized distributions $q(z_{ij})$.

Coordinate ascent variational inference in TSCBNs

Maximization of the ELBO of the factorized distributions $q(z_{ij})$ can be done using Coordinate Ascent Variational Inference, which uses iterative updates. By inserting the factorization of Eq. 33 in Eq. 29, it can be found that each updated variational distribution $q^*(z_{ij})$ is

$$q^*(z_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log p(Z, X)]\right) \tag{34}$$

Computation of expectation Any parameter of a node in the network (v or Δt) are updated by computing this exponential expectation iteratively until convergence. To compute those expectations efficiently the following assumptions are made:

- $\mathbb{E}$ is both computed over K observations and as stated above for all valid latent mappings from X_i to Z_i. Mathematically this results from Eqs. 27, while computationally this means to compute $\mathbb{E}$ over valid combinations only. This reduces computational costs.
- The computational costs are further optimized by enabling local computations per node. This is, as according to (Bishop et al. 2003) the expectation needs to only be computed over the Markov Blanket around the updated node z_{ij}.

Using those assumptions we can deduce update equations $q^*(z_{ij}^k)$ for state nodes v_{ij} and for temporal nodes Δt_{ij}. This is done by iteratively alternating between updating network parameters $p(Z|X)$ from all $q^*(z_{ij})$ and updating $q^*(z_{ij})$ from the current network parameters $p(Z|X)$. This is done as follows.

Update state nodes v_{ij}: With above assumptions and the definition of the TSCBN structure the update equation of any state node v_{ij} is given as

$$q^*(v_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log p(Z, X)]\right) \tag{35}$$

$$= \tag{36}$$

$$q^*(v_{ij}) \propto \exp\left(\mathbb{E}_{q_{-ij}}[\log\left(\Psi(v_{ij}) \cdot \Gamma(v_{ij}) \cdot \Xi(v_{ij})\right)]\right) \tag{37}$$

with Ψ as the factor with all parents of v_{ij}, Γ as all factors that contain co-parents of v_{ij} and Ξ as factor with children of v_{ij}, which is

$$\Psi(v_{ij}) = p(v_{ij}|Pa(v_{ij})) \tag{38}$$

$$\Gamma(v_{ij}) = \prod_{\substack{\gamma \in CoPa(v_{ij}) \\ \xi \in Ch(v_{ij})}} p(\xi|\gamma) \tag{39}$$

$$\Xi(v_{ij}) = \prod_{\xi \in Ch(v_{ij})} p(\xi|v_{ij} \cup CoPa(\xi)) \tag{40}$$

Notably, $\Xi(v_{ij})$ includes the observed sequence X_i with factor $p(X_i|v_{ij} \cup CoPa(v_{ij}))$, which was defined in Eq. 27 to be zero for invalid mappings.

Computation of $\mathbb{E}$: In this update equation all expectations are composed of terms that are solely discrete and terms that are of mixed type (i.e. continuous and discrete). This is, as all nodes in a TSCBN also have a temporal node that needs to be included in this step. For the purpose of clarity we split the expectation in a mixed $\mathbb{E}_c$ and a discrete part $\mathbb{E}_d$ which gives us the typical shape of the update equation

$$\begin{aligned}
\mathbb{E}[\log p(Z, X)] &= \mathbb{E}[\log(p(\Delta t_{ij}|v_{ij}, ...) + \log(p(\Delta t_{kr}|v_{ij}, ...) + ... && (41)\\
&\quad + \log(p(v_{ij}|v_{kr}, ...)] + \log(p(v_{xy}|v_{qs}, ...)] && (42)\\
&= \mathbb{E}[\log(p(\Delta t_{ij}|v_{ij}, ...) + \log(p(\Delta t_{kr}|v_{ij}, ...) + ...] && (43)\\
&\quad + \mathbb{E}[\log(p(v_{ij}|v_{kr}, ...)] + \log(p(v_{xy}|v_{qs}, ...)] = \mathbb{E}_c + \mathbb{E}_d && (44)
\end{aligned}$$

$\mathbb{E}_d$ can be simply computed by iteration of outcome combinations, while $\mathbb{E}_c$ is computed as follows. As all Δt are root nodes in TSCBNs, $\mathbb{E}_c$ is made up of components of the shape $\mathbb{E}[\Delta t|Pa(\Delta t)]$, with discrete nodes $Pa(\Delta t)$. Those components are computed as

$$\begin{aligned}
\mathbb{E}[\Delta t|Pa(\Delta t)] &= \sum_V (\int_{-\infty}^{\infty} \Delta t \cdot \log p(\Delta t|Pa(\Delta t)) \mathrm{d}\, \Delta\, \mathrm{t}) \cdot q(Pa(\Delta t)) && (45)\\
&= \sum_V \chi(\Delta t, Pa(\Delta t)) \cdot q(Pa(\Delta t)) && (46)
\end{aligned}$$

where V are all outcome combinations of the discrete parents of Δt, i.e. $Pa(\Delta t)$, and $q(Pa(\Delta t))$ its local probabilities. Notably, in CAVI the expectations are computed with respect to the update node, i.e. $\mathbb{E}_{q_{-ij}}$ is excluding $q(v_{ij})$.

Updating network parameters: With the local updated estimates the network parameters can be found with

$$p(v_{ij}|Pa(v_{ij})) = \prod_{\tau \in \{v_{ij}\} \cup Pa(v_{ij})} q(\tau) \tag{47}$$

This is possible according to the mean field assumption where we assume all nodes $q(v_{ij})$ to be pairwise independent.

Update temporal nodes Δt_{ij}: Here, for nodes Δt_{ij}, we still consider the structure in Fig. 4 and a Gaussian distribution per temporal node. With this, the update equation for the local estimate of Δt_{ij}, governed by its parameters μ and σ per observation is

$$q^*(\Delta t_{ij}; \mu^*_{ij}, \sigma^{*2}_{ij}) \propto \exp(\mathbb{E}_{\Delta t_{-ij}}[\log p(Z, X)]) \tag{48}$$

For this case again the structure of TSCBNs is assumed and only the Markov Blanket is considered per node, i.e. as Δt_{ij} are leaf nodes only one term with its parents is of relevance. Also, the valid mappings assumption in Eq. 27 holds. That gives the non constant part to compute the expectation over as

$$p(X, Z) = p(\Delta t_{ij}; \mu_{ij}, {\sigma_{ij}}^2 | Pa(\Delta t_{ij}))) \tag{49}$$

Here, this remaining term captures the distribution given its parents.
Estimating latent outcomes from observations: For the computation of the expectation per outcome combination and observation we map the absolute times of observed state changes to latent observed values for each node Δt_{ij} by using the method described in Sect. 4. This allows to get a full sample estimate of the network (with values for all Δt) for each outcome combination ω which is based on the current observation. With this latency in mappings from observed vectors X_i to its latent node outcomes is handled in time.
Per such estimated observation $\Delta t_{ij\text{obs}|\omega}$ we assume a Gaussian $\mathcal{N}(\mu_{ij|\omega} = \Delta t_{ij\text{obs}|\omega}, \sigma^2)$. The idea is to compute the expectation around the observed absolute times by interpolating them and by resolving latency via computation of the CAVI update around relevant parent outcomes. Also, we assume the variances ${\sigma_{ij}}^2$ to be fixed and only update each μ_{ij} per condition outcome combination ω that $Pa(\Delta t_{ij})$ can take.
Estimation of local node estimate $q^*(\Delta t_{ij}; \mu^*, {\sigma^*}^2)$: From the interpolated observations $\mu_{ij|\omega}$ and the current estimates of the outcomes $q(v_r = \omega_r)$ a local estimate per node is found with Eq. 48. For this the expectation can be written as

$$\begin{aligned}
&\mathbb{E}_{\Delta t_{-ij}}[\log p(Z, X)] && (50)\\
&= \sum_{\omega \in Pa(\Delta t_{ij})} \log p(\Delta t_{ij}; \mu_{ij}, {\sigma_{ij}}^2 | Pa(\Delta t_{ij}) = \omega) \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r) && (51)\\
&= \sum_{\omega \in Pa(\Delta t_{ij})} \log \mathcal{N}(\mu_{ij|\omega}, \sigma^2) \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r) && (52)
\end{aligned}$$

where v_r are all nodes of $Pa(\Delta t_{ij})$ with a specific outcome ω_r, e.g. for a node Δt_{12} v_r might correspond to nodes v_{12} and v_{11}. Also the Gaussian distribution is fixed with a mean of the given interpolated observation $\mu_{ij|\omega} = \Delta t_{ijobs|\omega}$.
When, inserting Eq. 52 into Eq. 48 it can be found that for the updated local estimate μ^*_{ij} of the temporal node Δt_{ij} it holds that

$$\mu^*_{ij} = \frac{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \mu_{ij|\omega} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)} \tag{53}$$

This makes intuitively sense, as the μ_{ij} governing Δt_{ij} is a weighted average over its outcomes depending on its likelihoods.

Updating network parameters: Finally the estimate $p(\Delta t_{ij}; \mu_{ij|\omega}, \sigma^2_{ij} | Pa(\Delta t_{ij})$ of each CPD of each temporal node can be found as the weighted average

$$\mu^*_{ij|\omega} = \frac{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \mu^*_{ij} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum\limits_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)} \tag{54}$$

Estimation algorithm With the given update equations the VI algorithm in Algorithm 2 is deduced. Notably, in this representation each line within the while loop consists of local updates per node that can be computed in parallel. Further, the observations X are included in the way described in this section at lines 3 and 6. ω resembles all possible parent outcome combinations.

Computing Δt

When computing any Δt_{ij} in a latent sequence (e.g. $\langle A, A, B, B \rangle$) from a full observation X_i (e.g. $\langle A, B \rangle$), in order to be usable in the computation of the expectations in CAVI each latent node requires a defined absolute time at which it occurs. We use the following interpolation strategy for this.

- Times between two elements are interpolated linearly within known parts, e.g. given A at $t = 1$ and B at $t = 2$ would result in times and values $\langle (A, 1), (A, 1.33), (A, 1.66), (B, 2.0) \rangle$ for a latent sequence $\langle A, A, A, B \rangle$.
- If the last observed element is followed by further elements in the latent sequence, its preceding element's distance is used for interpolation, e.g. having seen A at $t = 1$ and B at $t = 2$ that is extended to $\langle A, A, B, B, B \rangle$ would result in $\langle (A, 1), (A, 1.5), (B, 2), (B, 2.5), (B, 3.0) \rangle$. This extension time is determined by subtracting the last observed time by its predecessors interpolated time. If no predecessor is available the distance is computed by dividing the total time of the current sequence by its number of nodes. While this interpolation may potentially lead to temporal inconsistency, it produces a reasonable approximation that tends to decrease deviation from the true mean when enough sequences were observed.

Appendix D

Comparison of parameter estimation approaches

We also compared the proposed approaches estimation approaches against each other, with results shown in Fig. 12 (right) and in Fig. 13. In terms of runtime VI performs better than EM and MLE-R. This is, as expectations are computed directly within VI, while in both other approaches the MCMC sampling requires time. The expectation in VI is computed based on all allowed outcome combinations. Thus, with growing structure, combinations increase, which in turn leads to longer computational runtime. In EM and MLE-R bigger structures only increase the time required for sampling and computation of global estimates, which does increase less than the effect of growth in

Algorithm 2 CAVI Approach for parameter estimation in TSCBNs
Input: V: State Nodes, T: Temporal nodes, X: observations
Output: $\mu^*_{ij|\omega}$: Estimated parameters of temporal nodes, $p^*(v_{ij}|Pa(v_{ij}))$: Estimated Parameters of state nodes

```
1: $q(v_{ij}|\omega) = \mathcal{U}(); \forall v_{ij} \in V$    ▷ Initialization
2: while $\neg$ ($\mathcal{L}(q)$ converged) do
3:     $q^*(v_{ij}) = \exp(\mathbb{E}_{q_{-ij}}[\log(\Psi(v_{ij}) \cdot \Gamma(v_{ij}) \cdot \Xi(v_{ij}))]); \forall v_{ij} \in V$    ▷ State Node Updates
4:     $p^*(v_{ij}|Pa(v_{ij})) = mean(\prod_{\tau \in \{v_{ij}\} \cup Pa(v_{ij})} q^*(\tau)); \forall v_{ij} \in V$    ▷ State Network Updates
5:
6:     $\mu^*_{ij} = \frac{\sum_{\omega \in Pa(\Delta t_{ij})} \mu_{ij|\omega} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}; \forall \Delta t_{ij} \in T$    ▷ Temporal Node Updates
7:     $\mu^*_{ij|\omega} = \frac{\sum_{\omega \in Pa(\Delta t_{ij})} \mu^*_{ij} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}{\sum_{\omega \in Pa(\Delta t_{ij})} \cdot \prod_{\omega_r \in \omega} q(v_r = \omega_r)}; \forall \Delta t_{ij} \in T$    ▷ Temporal Network Updates
8: return $\mu^*_{ij|\omega} \forall \Delta t_{ij} \in T, p^*(v_{ij}|Pa(v_{ij})); \forall v_{ij} \in V$
```

combinations in VI. Thus, for larger network structures EM or MLE-R perform best, while for smaller networks VI is best. However, as can be seen in Fig. 13 (left), all approaches yield similar results in expressiveness, while MLE-R and EM yield slightly better results than VI. In contrast to MLE-R, EM and VI are more approximative approaches, which perform slightly worse. All approaches improve with decreasing SC probability, which makes sense, as with this less latency is present and the degree of approximation of latent variables is smaller. Figure 13 (mid) further shows that more samples yield better expressiveness for all approaches. This again is due to the increased amount of information available for estimation in that case. The temporal expressiveness is best in VI as its latent continuous nodes are included in the overall approximation procedure, which allows to improve those over time. For EM and MLE-R we used identical approaches, but the EM temporal approximation was updated on each iteration, while for MLE-R this estimate is computed in one run only. The effect of iteration seems to only deviate little from the estimate of one run, as Fig. 13 (right) shows, i.e. EM and MLE-R yield similar results. Overall, for the given networks VI performs better in runtime, is slightly less expressive in state, but most expressive in time. MLE-R and EM perform similar in those three categories and yield better expressiveness in state than VI. Consequently, all three approaches are about equally suited for parameter estimation in TSCBNs, while trading off expressiveness in state, expressiveness in time and runtime.

References

Aliferis C, Cooper G (1996) A structurally and temporally extended Bayesian belief network model: definitions, properties, and modeling techniques. In: Proceedings of the UAI. Morgan Kaufmann

Arroyo-Figueroa G, Sucar L (1999) A temporal Bayesian network for diagnosis and prediction. In: Proceedings of the UAI. Morgan Kaufmann

Arroyo-Figueroa G, Sucar L (2005) Temporal Bayesian network of events for diagnosis and prediction in dynamic domains. Appl Intell 23(2):66

Barber D (2012) Bayesian reasoning and machine learning. Cambridge University Press

Bartlett M, Cussens J (2013) Advances in Bayesian network learning using integer programming. arXiv preprint arXiv:1309.6825

Batal I, Sacchi (2009) Multivariate time series classification with temporal abstractions. In: FLAIRS conference

Berzuini C (1990) Representing time in causal probabilistic networks. In: Machine intelligence and pattern recognition, vol 10. Elsevier

Bhattacharjya D, Shanmugam K, Gao T, Mattei N, Varshney K, Subramanian D (2020) Event-driven continuous time Bayesian networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 3259–3266

Bishop CM, Spiegelhalter D, Winn J (2003) Vibes: a variational inference engine for Bayesian networks. In: Advances in neural information processing systems, pp 793–800

Bodon F (2005) A trie-based apriori implementation for mining frequent item sequences. In: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations. ACM, pp 56–65

Bubenzer J (2011) Minimization of acyclic dfas. In: Stringology, pp 132–146

Cooper GF, Herskovits E (1992) A Bayesian method for the induction of probabilistic networks from data. Mach Learn 9(4):309–347

Corneli M, Latouche P, Rossi F (2015) Modelling time evolving interactions in networks through a non stationary extension of stochastic block models. In: 2015 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE, pp 1590–1591

Daly R, Shen Q, Aitken S (2011) Learning Bayesian networks: approaches and issues. Knowl Eng Rev 26(2):99–157

De Campos LM, Fernández-Luna JM, Puerta JM (2003) An iterated local search algorithm for learning Bayesian networks with restarts based on conditional independence tests. Int J Intell Syst 18(2):221–235

De Carlo F, Borgia O, Tucci M (2013) Imperfect maintenance modelling by dynamic object oriented Bayesian networks. Int J Eng Technol 5(5):4282–4295

Dean T, Kanazawa K (1989) A model for reasoning about persistence and causation. Comput Intell 5(2):66

Galán S, Diez F (2000) Modeling dynamic causal interaction with Bayesian networks: temporal noisy gates. In: Proceedings 2nd international workshop on causal networks

Galán S, Diez F (2002) Networks of probabilistic events in discrete time. Int J Approx Reason 30(3):181–202

Gopalratnam K, Kautz H, Weld DS (2005) Extending continuous time Bayesian networks. In: Proceedings of the national conference on artificial intelligence. AAAI Press/MIT Press, Menlo Park, Cambridge, London, 1999, vol 20, p 981

Gu Y, Sun Y, Gao J (2017) The co-evolution model for social network evolving and opinion migration. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 175–184

Gunawardana A, Meek C (2016) Universal models of multivariate temporal point processes, pp 556–563

Hallac D, Vare S, Boyd S, Leskovec J (2017) Toeplitz inverse covariance-based clustering of multivariate time series data. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 215–223

Heckerman D, Geiger D, Chickering DM (1995) Learning Bayesian networks: the combination of knowledge and statistical data. Mach Learn 20(3):197–243

Karkera KR (2014) Building probabilistic graphical models with Python. Packt Publishing Ltd

Kwon WY, Suh IH (2012) A temporal Bayesian network with application to design of a proactive robotic assistant. In: 2012 IEEE international conference on robotics and automation. IEEE, pp 3685–3690

Leemans SJ, Fahland D, van der Aalst WM (2013) Discovering block-structured process models from event logs containing infrequent behaviour. In: International conference on business process management. Springer, pp 66–78

Mrowca A, Moser B, Günnemann S (2018a) Discovering groups of signals in in-vehicle network traces for redundancy detection and functional grouping. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 86–102

Mrowca A, Pramsohler T, Steinhorst S, Baumgarten U (2018b) Automated interpretation and reduction of in-vehicle network traces at a large scale. In: 2018 55th ACM/ESDA/IEEE design automation conference (DAC). IEEE, pp 1–6

Mrowca A, Nocker M, Steinhorst S, Günnemann S (2019) Learning temporal specifications from imperfect traces using Bayesian inference. In: 2019 56th ACM/ESDA/IEEE design automation conference (DAC), IEEE

Murphy KP, Russell S (2002) Dynamic Bayesian networks: representation, inference and learning

Nodelman U, Shelton CR, Koller D (2002) Continuous time Bayesian networks. In: Proceedings of the UAI. Morgan Kaufmann

Olesen K, Hejlesen O, Dessau R, Beltoft I, Trangeled M (2006) Diagnosing lyme disease-tailoring patient specific Bayesian networks for temporal reasoning. In: Probabilistic graphical models

Orphanou K, Stassopoulou A, Keravnou E (2016) Dbn-extended: a dynamic Bayesian network model extended with temporal abstractions for coronary heart disease prognosis. IEEE J Biomed Health Inform 20(3):944–952

Portinale L, Codetta-Raiteri D (2009) Generalizing continuous time Bayesian networks with immediate nodes. In: Proceedings of the workshop on graph structure for knowledge represetnation and reasoning, pp 12–17

Ryabov V, Trudel A (2004) Probabilistic temporal interval networks. In: Proceedings of the TIME 2004. IEEE

Savickas T, Vasilecas O (2014) Business process event log transformation into Bayesian belief network

Savickas T, Vasilecas O (2017) Decision support using belief network constructed from business process event log. Informatica 28(4):687–701

Scanagatta M, de Campos CP, Corani G, Zaffalon M (2015) Learning Bayesian networks with thousands of variables. In: Advances in neural information processing systems, pp 1864–1872

Schwarz G et al (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464

Shelton CR, Fan Y, Lam W, Lee J, Xu J (2010) Continuous time Bayesian network reasoning and learning engine. J Mach Learn Res 11:1137–1140

Simma A, Jordan MI (2012) Modeling events with cascades of Poisson processes. arXiv preprint arXiv:1203.3516

Song L, Kolar M, Xing EP (2009) Time-varying dynamic Bayesian networks. In: Advances in neural information processing systems, pp 1732–1740

Spirtes P, Glymour C (1991) An algorithm for fast recovery of sparse causal graphs. Soc Sci Comput Rev 9(1):62–72

Spirtes P, Glymour C, Scheines R (2000) Causation, prediction, and search. adaptive computation and machine learning

Sutrisnowati RA, Bae H, Park J, Ha BH (2013) Learning Bayesian network from event logs using mutual information test. In: 2013 IEEE 6th international conference on service-oriented computing and applications. IEEE, pp 356–360

Swaminathan S, Smidts C (1999) The event sequence diagram framework for dynamic probabilistic risk assessment. Reliab Eng Syst Saf 63(1):66

Tawfik A, Neufeld E (2000) Temporal reasoning and Bayesian networks. Comput Intell 16(3):66

Tsamardinos I, Brown LE, Aliferis CF (2006) The max-min hill-climbing Bayesian network structure learning algorithm. Mach Learn 65(1):31–78

Tucker A (2001) The automatic explanation of multivariate time series. PhD thesis, Ph. D. thesis, Birkbeck College, University of London, UK

Van den Broeck G, Mohan K, Choi A, Darwiche A, Pearl J (2015) Efficient algorithms for Bayesian network parameter learning from incomplete data. Meila, Marina

Van der Aalst W, Weijters T, Maruster L (2004) Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142

Walicki M, Ferreira DR (2011) Sequence partitioning for process mining with unlabeled event logs. Data Knowl Eng 70(10):821–41

Weijters A, van Der Aalst WM, De Medeiros AA (2006) Process mining with the heuristics miner-algorithm, vol 166. Technische Universiteit Eindhoven, Tech Rep WP, pp 1–34

Yuan C, Malone B (2013) Learning optimal Bayesian networks: a shortest path perspective. J Artif Intell Res 48:23–65