Analyzing complex systems with cascades using continuous-time Bayesian networks

TIME2023 - September 25, 2023

Alessandro Bregoli[1], Karin Rathsman[3], Marco Scutari[4], Fabio Stella[1], Søren Wengel Mogensen[2]

[1]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Italy
[2]Department of Automatic Control, Lund University
[3]European Spallation Source ERIC, Lund, Sweden
[4]Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.

Commonly those phenomena have a **normal behavior** and an **abnormal one**.

The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.

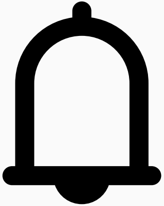
Introduction

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.

Commonly those phenomena have a **normal behavior** and an **abnormal one**.

The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.



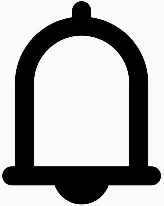
Introduction

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.

Commonly those phenomena have a **normal behavior** and an **abnormal one**.

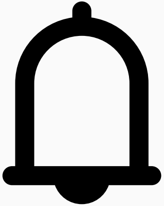
The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.



Introduction

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.



Commonly those phenomena have a **normal behavior** and an **abnormal one**.

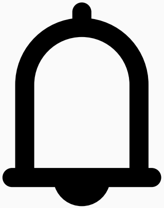
The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.

Introduction

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.

Commonly those phenomena have a **normal behavior** and an **abnormal one**.

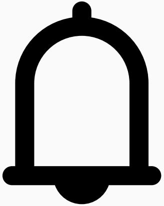


The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.

Introduction

Many real world phenomena can be seen as **sequence of events** such as:

- Monitoring vital signs of a patient (healthcare).
- System log (operating system).
- Monitoring of industrial processes.

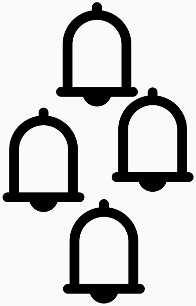


Commonly those phenomena have a **normal behavior** and an **abnormal one**.

The ability to **identify behavioral change** can enable **early intervention** before a critical scenario occurs.

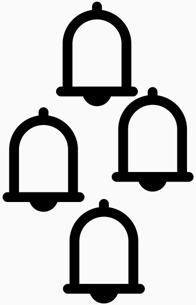
Alarms

- The **problem** we want to address is to **extract knowledge** from a **sequence of events**.
- We will focus on **binary events** such as **alarms**.
- The **goal** of this work is to identify the beginning of **alarm cascades** (*Rapid sequence of events*)
- The **method** we proposed is based on **Continuous Time Bayesian Network**



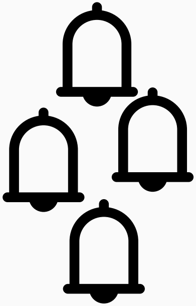
Alarms

- The **problem** we want to address is to **extract knowledge** from a **sequence of events**.
- We will focus on **binary events** such as **alarms**.
- The **goal** of this work is to identify the beginning of **alarm cascades** (*Rapid sequence of events*)
- The **method** we proposed is based on **Continuous Time Bayesian Network**



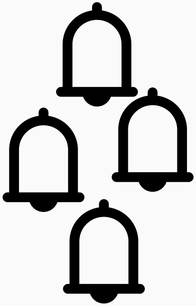
Alarms

- The **problem** we want to address is to **extract knowledge** from a **sequence of events**.
- We will focus on **binary events** such as **alarms**.
- The **goal** of this work is to identify the beginning of **alarm cascades** (*Rapid sequence of events*)
- The **method** we proposed is based on **Continuous Time Bayesian Network**



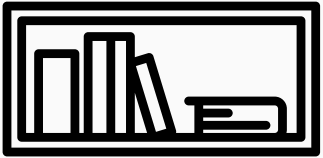
Alarms

- The **problem** we want to address is to **extract knowledge** from a **sequence of events**.
- We will focus on **binary events** such as **alarms**.
- The **goal** of this work is to identify the beginning of **alarm cascades** (*Rapid sequence of events*)
- The **method** we proposed is based on **Continuous Time Bayesian Network**



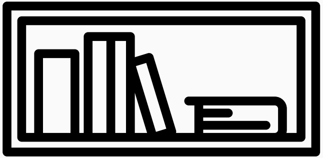
Outline

- Continuous Time Markov Process
- Continuous Time Bayesian Network
- Sentry State
- Synthetic Experiments



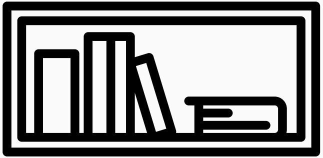
Outline

- Continuous Time Markov Process
- Continuous Time Bayesian Network
- Sentry State
- Synthetic Experiments



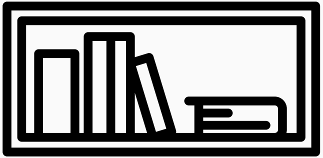
Outline

- Continuous Time Markov Process
- Continuous Time Bayesian Network
- Sentry State
- Synthetic Experiments



Outline

- Continuous Time Markov Process
- Continuous Time Bayesian Network
- Sentry State
- Synthetic Experiments



Continuous Time Markov Process

Formal Description

A Continuous Time Markov Process (CTMP)[1] is a **continuous time stochastic process**:

$$X = \{X(t_i) : t_i \in [0, \infty), t_{i-1} < t_i\}$$

which satisfies the **Markov Property**:

$$X(t_1) \perp\!\!\!\perp X(t_3) | X(t_2), \forall t_1 < t_2 < t_3$$

The **state** of a CTMP **changes** in continuous time and can take value over a discrete set or domain $x \in Val(X)$.

[1] C. R. Shelton and G. Ciardo, **"Tutorial on structured continuous-time markov processes,"** *Journal of Artificial Intelligence Research*, vol. 51, pp. 725–778, 2014.

Formal Description

A Continuous Time Markov Process (CTMP)[1] is a **continuous time stochastic process**:

$$X = \{X(t_i) : t_i \in [0, \infty), t_{i-1} < t_i\}$$

which satisfies the **Markov Property**:

$$X(t_1) \perp\!\!\!\perp X(t_3) | X(t_2), \forall t_1 < t_2 < t_3$$

The **state** of a CTMP **changes** in continuous time and can take value over a discrete set or domain $x \in Val(X)$.

[1] C. R. Shelton and G. Ciardo, **"Tutorial on structured continuous-time markov processes,"** *Journal of Artificial Intelligence Research*, vol. 51, pp. 725–778, 2014.

Formal Description

A Continuous Time Markov Process (CTMP)[1] is a **continuous time stochastic process**:

$$X = \{X(t_i) : t_i \in [0, \infty), t_{i-1} < t_i\}$$

which satisfies the **Markov Property**:

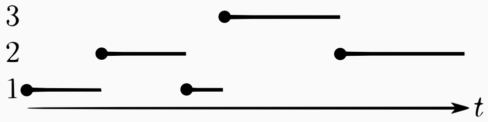
$$X(t_1) \perp\!\!\!\perp X(t_3) | X(t_2), \forall t_1 < t_2 < t_3$$

The **state** of a CTMP **changes** in continuous time and can take value over a discrete set or domain $x \in Val(X)$.

[1] C. R. Shelton and G. Ciardo, **"Tutorial on structured continuous-time markov processes,"** *Journal of Artificial Intelligence Research*, vol. 51, pp. 725–778, 2014.

Trajectory

A realization of the CTMP is a **trajectory**; a right-continuous piece-wise constant function over time that can be represented as a sequence of time-indexed events:



$$\sigma = \{\langle t_0, X(t_0)\rangle, \langle t_1, X(t_1)\rangle, ..., \langle t_I, X(t_I)\rangle\}, \qquad t_0 < t_1 < \cdots < t_I$$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \ \forall \ i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameters

A CTMP can be parameterized as follows:

- An **initial distribution** $P(X(0))$. *It describes the process at time* $t = 0$
- An **intensity matrix** Q_X. *It models the evolution of* X *through time*

$$Q_X = \begin{bmatrix} -q_1 & q_{12} & q_{13} \\ q_{21} & -q_2 & q_{23} \\ q_{31} & q_{32} & -q_3 \end{bmatrix} \qquad q_i > 0, q_{ij} \geqslant 0 \; \forall \, i, j$$

Each row of Q_X **sums up to 0** and models two processes:

- An exponential distribution with parameter $q_i \in \mathbb{R}^+$
- A multinomial distribution with parameters $\theta_{ij} = \frac{q_{ij}}{q_i}$

Parameter Learning - Maximum Likelihood Approach

The parameters can be estimated with the Maximum Likelihood Approach as follows:

$$\hat{q}_{i,j} = \frac{N[i,j]}{T[i]}, \qquad \hat{q}_i = \sum_{i \neq j} \frac{N[i,j]}{T[i]}$$

where N and T are the two sufficient statistics:

- $N[i,j]$: number of transitions from state i to state j.
- $T[i]$: time spent in state i

Continuous Time Bayesian Network

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \ldots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \ldots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \ldots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \dots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \dots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2]U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \dots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

Formal Description

A Continuous Time Bayesian Network (CTBN)[2] is an extension of CTMP capable of dealing with a **factored state space**:

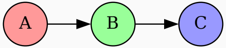
$$\mathcal{N} = \langle P_0, \mathbf{X}, \mathcal{G}, \mathbf{Q_X} \rangle$$

- An **initial distribution** P_0.
- A set of L variables $\mathbf{X} = \{X_1, X_2, \dots, X_L\}$
- Each varible $X_i \in \mathbf{X}$ changes in continuous time and can take value over a discrete set or domain $x \in Val(X_i)$.
- $Val(\mathbf{X}) = Val(X_1) \times Val(X_2) \times \cdots \times Val(X_L)$
- A directed, possibly cyclic, graph $\mathcal{G}$.
- A set of **Conditional Intensity Matrices** (CIM) $\mathbf{Q_X}$.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation,

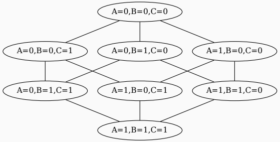
CTBN - Example

Dependency Graph



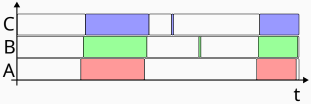
It represents the structure of the network with 3 nodes ($\mathcal{G}$).

State space graph



It describes the state space and all the possible transitions from one state to another ($\mathcal{G}_s$).

Trajectory



It is an example of trajectory for the network; the white spaces indicate a variable in state 0 and coloured spaces indicate a variable in state 1.

Parameter Learning - Maximum Likelihood Approach

The parameters for a variable $X_k \in \mathbf{X}$ with a specific configuration of the parent-set $\mathbf{u}$ can be estimated with the Maximum Likelihood Approach as follows:

$$\hat{q}_{i,j|\mathbf{u}} = \frac{N\left[i, j|\mathbf{u}\right]}{T\left[i|\mathbf{u}\right]}, \qquad \hat{q}_{i|\mathbf{u}} = \sum_{i \neq j} \frac{N\left[i, j|\mathbf{u}\right]}{T\left[i|\mathbf{u}\right]}$$

where N and T are the two sufficient statistics:

- $N\left[i, j|\mathbf{u}\right]$: number of transitions from state i to state j given the parent-set.
- $T\left[i|\mathbf{u}\right]$: time spent in state i given the parent-set.

Structure Learning

There are two main approaches to learn the structure of a CTBN:

- **Score Based Approach**: a Bayesian score is used to evaluate and compare different candidate structures, a search algorithm is used to find the structure that achieves the highest score[3].
- **Constraint Based Approach**: a set of hypothesis is formulated and specific tests are applied to assess the independence between variables, then an algorithm is developed to efficiently apply Hypothesis Testing[4].

[3] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation, Stanford University, 2007.

[4] A. Bregoli, M. Scutari, and F. Stella, **"A constraint-based algorithm for the structural learning of continuous-time bayesian networks,"** *International Journal of Approximate Reasoning*, vol. 138, pp. 105–122, 2021.

Structure Learning

There are two main approaches to learn the structure of a CTBN:

- **Score Based Approach**: a Bayesian score is used to evaluate and compare different candidate structures, a search algorithm is used to find the structure that achieves the highest score[3].
- **Constraint Based Approach**: a set of hypothesis is formulated and specific tests are applied to assess the independence between variables, then an algorithm is developed to efficiently apply Hypothesis Testing[4].

[3] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation, Stanford University, 2007.

[4] A. Bregoli, M. Scutari, and F. Stella, **"A constraint-based algorithm for the structural learning of continuous-time bayesian networks,"** *International Journal of Approximate Reasoning*, vol. 138, pp. 105–122, 2021.

Structure Learning

There are two main approaches to learn the structure of a CTBN:

- **Score Based Approach**: a Bayesian score is used to evaluate and compare different candidate structures, a search algorithm is used to find the structure that achieves the highest score[3].
- **Constraint Based Approach**: a set of hypothesis is formulated and specific tests are applied to assess the independence between variables, then an algorithm is developed to efficiently apply Hypothesis Testing[4].

[3] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation, Stanford University, 2007.

[4] A. Bregoli, M. Scutari, and F. Stella, **"A constraint-based algorithm for the structural learning of continuous-time bayesian networks,"** *International Journal of Approximate Reasoning*, vol. 138, pp. 105–122, 2021.

Reward Function

A **reward function** is a function that maps values of one or more variables onto a real number. Such a function can be introduced for both CTMPs and CTBNs. Intuitively, the reward function represents two quantities:

- $\mathcal{R}(x) : Val(\mathbf{X}) \rightarrow \mathbb{R}$; the instantaneous reward of state x
- $\mathcal{C}(x, x') : Val(\mathbf{X}) \times Val(\mathbf{X}) \rightarrow \mathbb{R}$; the lump sum reward when $\mathbf{X}$ transitions from state x to state x'.

It is possible to adapt the reward function for evaluating the evolution of the process. Specifically, we use the lump sum reward function as an indicator of transitions:

$$\mathcal{C}(x, x') = \begin{cases} 1 & \text{if } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

Reward Function

A **reward function** is a function that maps values of one or more variables onto a real number. Such a function can be introduced for both CTMPs and CTBNs. Intuitively, the reward function represents two quantities:

- $\mathcal{R}(x) : Val(\mathbf{X}) \rightarrow \mathbb{R}$; the instantaneous reward of state x
- $\mathcal{C}(x, x') : Val(\mathbf{X}) \times Val(\mathbf{X}) \rightarrow \mathbb{R}$; the lump sum reward when $\mathbf{X}$ transitions from state x to state x'.

It is possible to adapt the reward function for evaluating the evolution of the process. Specifically, we use the lump sum reward function as an indicator of transitions:

$$\mathcal{C}(x, x') = \begin{cases} 1 & \text{if } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

Reward Function

A **reward function** is a function that maps values of one or more variables onto a real number. Such a function can be introduced for both CTMPs and CTBNs. Intuitively, the reward function represents two quantities:

- $\mathcal{R}(x) : Val(\mathbf{X}) \to \mathbb{R}$; the instantaneous reward of state x
- $\mathcal{C}(x, x') : Val(\mathbf{X}) \times Val(\mathbf{X}) \to \mathbb{R}$; the lump sum reward when $\mathbf{X}$ transitions from state x to state x'.

It is possible to adapt the reward function for evaluating the evolution of the process. Specifically, we use the lump sum reward function as an indicator of transitions:

$$\mathcal{C}(x, x') = \begin{cases} 1 & \text{if } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

Reward Function

A **reward function** is a function that maps values of one or more variables onto a real number. Such a function can be introduced for both CTMPs and CTBNs. Intuitively, the reward function represents two quantities:

- $\mathcal{R}(x) : Val(\mathbf{X}) \rightarrow \mathbb{R}$; the instantaneous reward of state x
- $\mathcal{C}(x, x') : Val(\mathbf{X}) \times Val(\mathbf{X}) \rightarrow \mathbb{R}$; the lump sum reward when $\mathbf{X}$ transitions from state x to state x'.

It is possible to adapt the reward function for evaluating the evolution of the process. Specifically, we use the lump sum reward function as an indicator of transitions:

$$\mathcal{C}(x, x') = \begin{cases} 1 & \text{if } x \neq x' \\ 0 & \text{otherwise} \end{cases}$$

Reward Evaluation

Two options are available for computing the expected reward:

- *finite-horizon expected reward*:

$$V_{t_I}(x) = \mathbb{E}\left[\sum_{i=0}^{I-1} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} \mathcal{R}(\mathbf{X}(t_i))dt\right]$$

- *infinite-horizon expected discounted reward*:

$$V_{\alpha}(x) = \mathbb{E}\left[\sum_{i=0}^{\infty} e^{-\alpha t_i}\mathbf{X}(X(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\alpha t}\mathcal{R}(\mathbf{X}(t_i))dt\right]$$

Reward Evaluation

Two options are available for computing the expected reward:

- *finite-horizon expected reward*:

$$V_{t_I}(x) = \mathbb{E}\left[\sum_{i=0}^{I-1} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} \mathcal{R}(\mathbf{X}(t_i))dt\right]$$

- *infinite-horizon expected discounted reward*:

$$V_\alpha(x) = \mathbb{E}\left[\sum_{i=0}^{\infty} e^{-\alpha t_i} \mathbf{X}(X(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\alpha t} \mathcal{R}(\mathbf{X}(t_i))dt\right]$$

Reward Evaluation

Two options are available for computing the expected reward:

- *finite-horizon expected reward*:

$$V_{t_I}(x) = \mathbb{E}\left[\sum_{i=0}^{I-1} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} \mathcal{R}(\mathbf{X}(t_i))dt\right]$$

- *infinite-horizon expected discounted reward*:

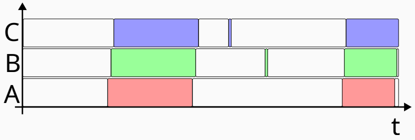
$$V_{\alpha}(x) = \mathbb{E}\left[\sum_{i=0}^{\infty} e^{-\alpha t_i}\mathbf{X}(X(t_i), \mathbf{X}(t_{i+1})) + \int_{t_i}^{t_{i+1}} e^{-\alpha t}\mathcal{R}(\mathbf{X}(t_i))dt\right]$$

Sentry State

Description

We informally defined a **sentry state** as a **state** of the CTBN which triggers a **ripple effect** , i.e., it triggers a *fast sequence of events* to occur due to fast and subsequent state changes.

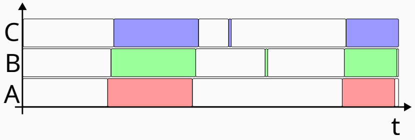
- This **informal definition** does **not** allow us to **classify** a state as **sentry state**.
- The best we can do with this definition is to **order** the states from the most likely sentry state to the less likely sentry state.



Description

We informally defined a **sentry state** as a **state** of the CTBN which triggers a **ripple effect** , i.e., it triggers a *fast sequence of events* to occur due to fast and subsequent state changes.

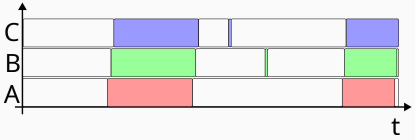
- This **informal definition** does **not** allow us to **classify** a state as **sentry state**.
- The best we can do with this definition is to **order** the states from the most likely sentry state to the less likely sentry state.



Description

We informally defined a **sentry state** as a **state** of the CTBN which triggers a **ripple effect** , i.e., it triggers a *fast sequence of events* to occur due to fast and subsequent state changes.

- This **informal definition** does **not** allow us to **classify** a state as **sentry state**.
- The best we can do with this definition is to **order** the states from the most likely sentry state to the less likely sentry state.



Cascade identification - Naive Approach

A **cascade of events** is a fast sequence of transitions, where fast is **relative** to the rest of the transitions observed.

Cascade identification can be carried out with the following naive approach:

- Identify λ_{ft}: determines when a **transition** is considered **fast**.
- Identify λ_{mcl}: the **minimum cascade length** determines the minimum number of successive fast events to be considered a cascade.
- Identifying in the trajectory subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft}.

Cascade identification - Naive Approach

A **cascade of events** is a fast sequence of transitions, where fast is **relative** to the rest of the transitions observed.

Cascade identification can be carried out with the following naive approach:

- Identify λ_{ft}: determines when a **transition** is considered **fast**.
- Identify λ_{mcl}: the **minimum cascade length** determines the minimum number of successive fast events to be considered a cascade.
- Identifying in the trajectory subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft}.

Cascade identification - Naive Approach

A **cascade of events** is a fast sequence of transitions, where fast is **relative** to the rest of the transitions observed.

Cascade identification can be carried out with the following naive approach:

- Identify λ_{ft}: determines when a **transition** is considered **fast**.
- Identify λ_{mcl}: the **minimum cascade length** determines the minimum number of successive fast events to be considered a cascade.
- Identifying in the trajectory subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft}.

Cascade identification - Naive Approach

A **cascade of events** is a fast sequence of transitions, where fast is **relative** to the rest of the transitions observed.

Cascade identification can be carried out with the following naive approach:

- Identify λ_{ft}: determines when a **transition** is considered **fast**.
- Identify λ_{mcl}: the **minimum cascade length** determines the minimum number of successive fast events to be considered a cascade.
- Identifying in the trajectory subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft}.

Cascade identification - Naive Approach

A **cascade of events** is a fast sequence of transitions, where fast is **relative** to the rest of the transitions observed.

Cascade identification can be carried out with the following naive approach:

- Identify λ_{ft}: determines when a **transition** is considered **fast**.
- Identify λ_{mcl}: the **minimum cascade length** determines the minimum number of successive fast events to be considered a cascade.
- Identifying in the trajectory subsets of consecutive transitions with length at least λ_{mcl} and with a transition time between each pair of consecutive events of less than λ_{ft}.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begin**s.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begin**s.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begins**.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begin**s.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begin**s.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Sentry State - Naive Approach

The **cascade identification** can also be used to recognize **sentry state**.

Once a cascade of events has been identified, the **sentry state** is the state from which the **cascade begin**s.

To order the states from the most probable sentry state to the least probable sentry state, two quantities were defined:

- *Naive Count*: the number of times a state starts a cascade.
- *Naive Score*: the fraction of times that observing a specific state coincides with the start of a cascade.

The main limitation of this approach is the difficulty of identifying the correct parameters as it requires knowing in advance **common durations** and **sizes of event cascades**.

Expected Discounted Number of Transitions

- Our goal consists in identifying a **sentry state** overcoming the limitations of the naive approach.
- Starting from the informal definition in the previous slide we want to **define a heuristic** to discover sentry states.
- The simplest heuristic is the **Expected Discounted Number of Transitions** estimated for each state.

$$EDNT_{\alpha}(x) = E\left[\sum_{i=0}^{\infty} e^{-\alpha t_i} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1}))\right]$$

Expected Discounted Number of Transitions

- Our goal consists in identifying a **sentry state** overcoming the limitations of the naive approach.
- Starting from the informal definition in the previous slide we want to **define a heuristic** to discover sentry states.
- The simplest heuristic is the **Expected Discounted Number of Transitions** estimated for each state.

$$EDNT_\alpha(x) = E\left[\sum_{i=0}^{\infty} e^{-\alpha t_i} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1}))\right]$$

Expected Discounted Number of Transitions

- Our goal consists in identifying a **sentry state** overcoming the limitations of the naive approach.
- Starting from the informal definition in the previous slide we want to **define a heuristic** to discover sentry states.
- The simplest heuristic is the **Expected Discounted Number of Transitions** estimated for each state.

$$EDNT_\alpha(x) = E\left[\sum_{i=0}^{\infty} e^{-\alpha t_i} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1}))\right]$$

Expected Discounted Number of Transitions

- Our goal consists in identifying a **sentry state** overcoming the limitations of the naive approach.
- Starting from the informal definition in the previous slide we want to **define a heuristic** to discover sentry states.
- The simplest heuristic is the **Expected Discounted Number of Transitions** estimated for each state.

$$EDNT_{\alpha}(x) = E\left[\sum_{i=0}^{\infty} e^{-\alpha t_i} C(\mathbf{X}(t_i), \mathbf{X}(t_{i+1}))\right]$$

Relative Expected Discounted Number of Transitions

- The **EDNT** is capable of identifying states with a high number of expected transitions.
- However, the EDNT is not capable of identifying a sentry state.
- For this reason we introduced the **Relative Expected Discounted Number of Transitions** REDNT capable of taking into account the number of transitions for the **neighborhood** of a state.

$$REDNT_{\alpha}(x) = \max_{x' \in \mathrm{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_{\alpha}(x)}{EDNT_{\alpha}(x')}$$

Relative Expected Discounted Number of Transitions

- The **EDNT** is capable of identifying states with a high number of expected transitions.
- However, the EDNT is not capable of identifying a sentry state.
- For this reason we introduced the **Relative Expected Discounted Number of Transitions** REDNT capable of taking into account the number of transitions for the **neighborhood** of a state.

$$REDNT_{\alpha}(x) = \max_{x' \in \mathrm{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_{\alpha}(x)}{EDNT_{\alpha}(x')}$$

Relative Expected Discounted Number of Transitions

- The **EDNT** is capable of identifying states with a high number of expected transitions.
- However, the EDNT is not capable of identifying a sentry state.
- For this reason we introduced the **Relative Expected Discounted Number of Transitions** REDNT capable of taking into account the number of transitions for the **neighborhood** of a state.

$$REDNT_{\alpha}(x) = \max_{x' \in \mathrm{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_{\alpha}(x)}{EDNT_{\alpha}(x')}$$

Relative Expected Discounted Number of Transitions

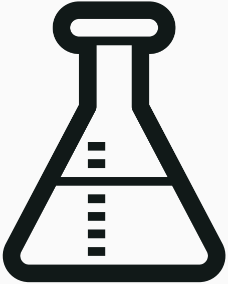
- The **EDNT** is capable of identifying states with a high number of expected transitions.
- However, the EDNT is not capable of identifying a sentry state.
- For this reason we introduced the **Relative Expected Discounted Number of Transitions** REDNT capable of taking into account the number of transitions for the **neighborhood** of a state.

$$REDNT_\alpha(x) = \max_{x' \in \mathrm{Ne}_{\mathcal{G}_s}(x)} \frac{EDNT_\alpha(x)}{EDNT_\alpha(x')}$$

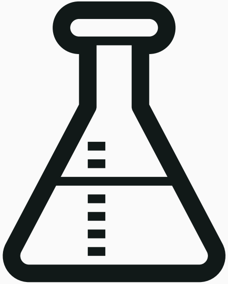
Synthetic experiments

Introduction

- In this section we present some **synthetic experiments**.
- Each experiment contains **one sentry state**.
- We distinguish between **slow nodes** and **fast nodes**.
- We will take into account only the **most likely sentry state** identified by **REDNT**.

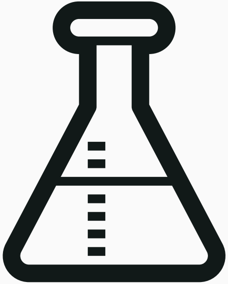


- In this section we present some **synthetic experiments**.
- Each experiment contains **one sentry state**.
- We distinguish between **slow nodes** and **fast nodes**.
- We will take into account only the **most likely sentry state** identified by **REDNT**.

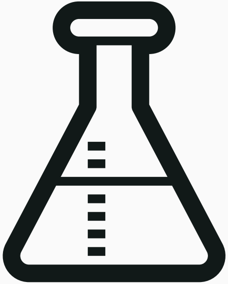


Introduction

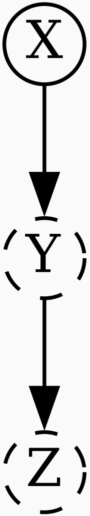
- In this section we present some **synthetic experiments**.
- Each experiment contains **one sentry state**.
- We distinguish between **slow nodes** and **fast nodes**.
- We will take into account only the **most likely sentry state** identified by **REDNT**.



- In this section we present some **synthetic experiments**.
- Each experiment contains **one sentry state**.
- We distinguish between **slow nodes** and **fast nodes**.
- We will take into account only the **most likely sentry state** identified by **REDNT**.



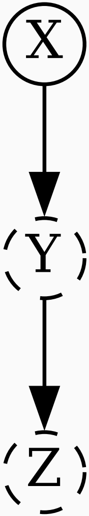
Chain

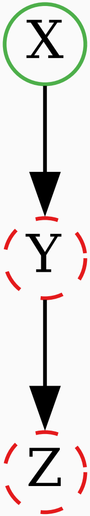


Legend

- **Continuous Line**: Slow Node
- **Dashed Line**: Fast Node
- **Green Line**: Node set to 1 (Alarm On)
- **Red Line**: Node set to 0 (Alarm off)

Chain

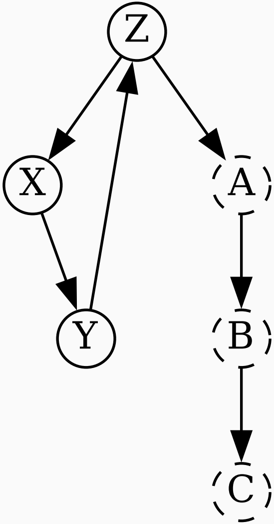




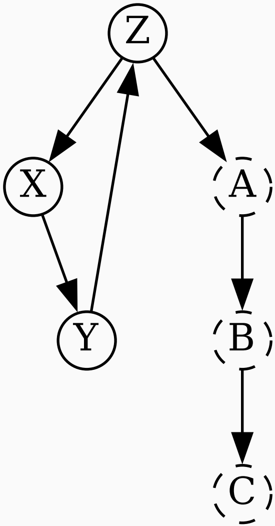
Legend

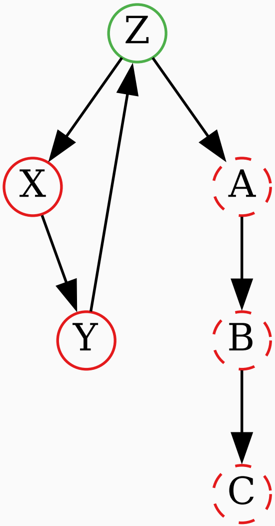
- **Continuous Line**: Slow Node
- **Dashed Line**: Fast Node
- **Green Line**: Node set to 1 (Alarm On)
- **Red Line**: Node set to 0 (Alarm off)

Cycle + Chain

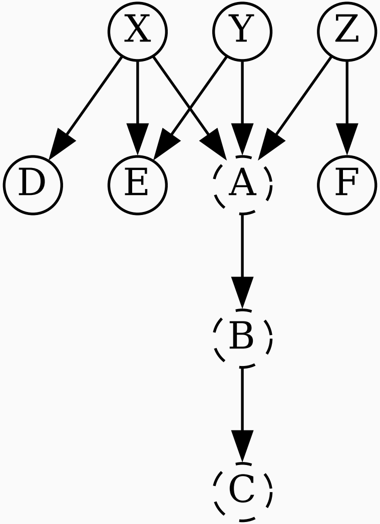


Cycle + Chain

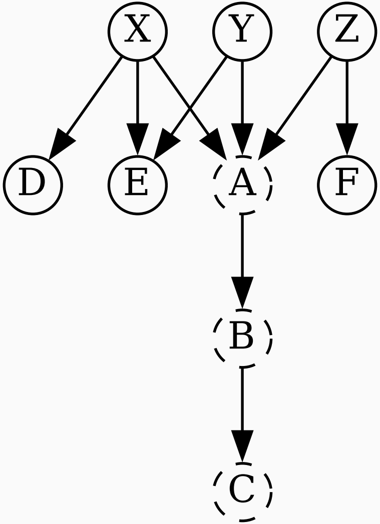


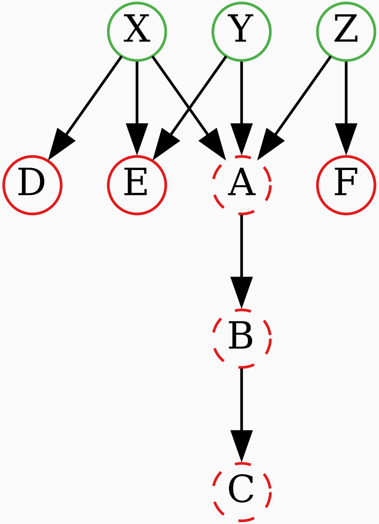


9 nodes



9 nodes





Comparison between Naive approach and REDNT: Jaccard similarity

- Since the experiments are synthetic, the parameters λ_{ft} and λ_{mcl} were easily identified.
- We used the *Naive Score* to order the states for the naive approach.

- We compared the Naive Score with the REDNT.
- We used the Jaccard similarity to compare the two approaches.

Comparison between Naive approach and REDNT: Jaccard similarity

- Since the experiments are synthetic, the parameters λ_{ft} and λ_{mcl} were easily identified.
- We used the *Naive Score* to order the states for the naive approach.

- We compared the Naive Score with the REDNT.
- We used the Jaccard similarity to compare the two approaches.

Comparison between Naive approach and REDNT: Jaccard similarity

- Since the experiments are synthetic, the parameters λ_{ft} and λ_{mcl} were easily identified.
- We used the *Naive Score* to order the states for the naive approach.

- We compared the Naive Score with the REDNT.
- We used the Jaccard similarity to compare the two approaches.

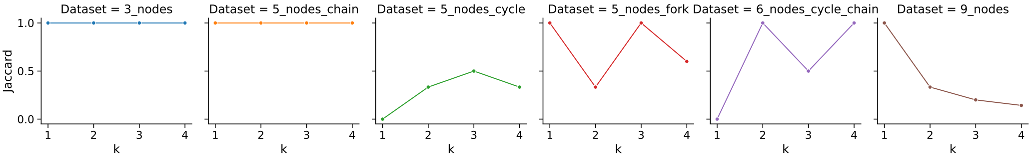
Comparison between Naive approach and REDNT: Jaccard similarity

- Since the experiments are synthetic, the parameters λ_{ft} and λ_{mcl} were easily identified.
- We used the *Naive Score* to order the states for the naive approach.

- We compared the Naive Score with the REDNT.
- We used the Jaccard similarity to compare the two approaches.

Comparison between Naive approach and REDNT: Jaccard similarity

- Since the experiments are synthetic, the parameters λ_{ft} and λ_{mcl} were easily identified.
- We used the *Naive Score* to order the states for the naive approach.
- We compared the Naive Score with the REDNT.
- We used the Jaccard similarity to compare the two approaches.



Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Conclusions

The **REDNT** is capable of finding **sentry states** in all the synthetic examples.

Pros

- Simple implementation
- Interaction with expert knowledge
- Easier hyper-parameter selection (compared the the naive approach)

Cons

- It requires us to explore the entire state space
- Hyper-parameter tuning
- Difficult to identify Sentry states with a small change in speed.

Future work

The proposed approach can be extended in many ways:

- Identify the **subset of nodes** that causes the **ripple effect** in a **sentry state**.
- Define **new metrics** that **do not require us to explore the entire state space**.

Future work

The proposed approach can be extended in many ways:

- Identify the **subset of nodes** that causes the **ripple effect** in a **sentry state**.
- Define **new metrics** that **do not require us to explore the entire state space**.

Future work

The proposed approach can be extended in many ways:

- Identify the **subset of nodes** that causes the **ripple effect** in a **sentry state**.
- Define **new metrics** that **do not require us to explore the entire state space**.

Thank You for your Attention

References i

[1] C. R. Shelton and G. Ciardo, **"Tutorial on structured continuous-time markov processes,"** *Journal of Artificial Intelligence Research*, vol. 51, pp. 725–778, 2014.

[2] U. D. Nodelman, **"Continuous time bayesian networks,"** Ph.D. dissertation, Stanford University, 2007.

[3] A. Bregoli, M. Scutari, and F. Stella, **"A constraint-based algorithm for the structural learning of continuous-time bayesian networks,"** *International Journal of Approximate Reasoning*, vol. 138, pp. 105–122, 2021.