The control problem of probabilistic Boolean control networks

Carmen Del Vecchio

in collaboration with Amol Yerudkar[1], Antonio Acernese[2], and Luigi Glielmo[3]

CDC Workshop 2023 – Semi-Tensor Product of Matrices and Its Applications

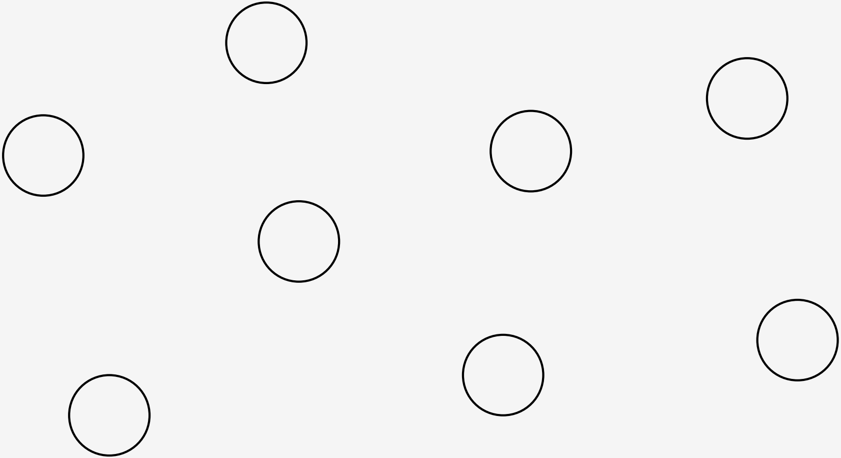
December 12, 2023

[1]Zhejiang Normal University, Jinhua, PR China; [2]Stellantis N.V.; [3]University of Federico II, Naples, Italy

Contents

Contents

Probabilistic Boolean network



Probabilistic Boolean network[2]

- The evolution of a probabilistic Boolean network (PBN) is determined by a set of n logic first order difference equations:

$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \qquad\qquad \vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \end{cases}$$

[2] D. Cheng, H. Qi, and Z. Li (2010). *Analysis and Control of Boolean Networks: A Semi-tensor Product Approach*. Communications and Control Engineering. Springer London.

Probabilistic Boolean network[2]

- The evolution of a probabilistic Boolean network (PBN) is determined by a set of n logic first order difference equations:

$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \qquad\qquad\vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \end{cases}$$

$\mathcal{X}(t) \in \mathcal{B}^n$
$f_i \in \{f_i^1, f_i^2, \ldots, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \rightarrow \mathcal{B}$, and with $Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, $\gamma_i \in \{1, 2, \ldots, l_i\}$, $i \in \{1, 2, \ldots, n\}$, and $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$

[2] D. Cheng, H. Qi, and Z. Li (2010). *Analysis and Control of Boolean Networks: A Semi-tensor Product Approach*. Communications and Control Engineering. Springer London.

Probabilistic Boolean network[2]

- The evolution of a probabilistic Boolean network (PBN) is determined by a set of n logic first order difference equations:

$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \\ \qquad\qquad\vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t)\big) \end{cases}$$

  $\mathcal{X}(t) \in \mathcal{B}^n$
  $f_i \in \{f_i^1, f_i^2, \ldots, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \to \mathcal{B}$, and with $Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, $\gamma_i \in \{1, 2, \ldots, l_i\}$, $i \in \{1, 2, \ldots, n\}$, and $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$
- There are $N = \prod_{i=1}^{n} l_i$ possible realizations of the network.

[2] D. Cheng, H. Qi, and Z. Li (2010). *Analysis and Control of Boolean Networks: A Semi-tensor Product Approach*. Communications and Control Engineering. Springer London.

Probabilistic Boolean network[2]

- The evolution of a probabilistic Boolean network (PBN) is determined by a set of n logic first order difference equations:

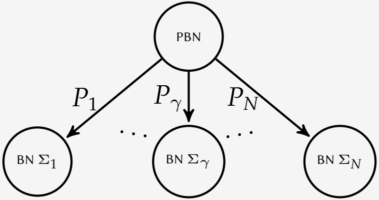
$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \dots, \mathcal{X}_n(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \dots, \mathcal{X}_n(t)\big) \\ \qquad\qquad \vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \dots, \mathcal{X}_n(t)\big) \end{cases}$$

  $\mathcal{X}(t) \in \mathcal{B}^n$
  $f_i \in \{f_i^1, f_i^2, \dots, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \to \mathcal{B}$, and with $Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, $\gamma_i \in \{1, 2, \dots, l_i\}$, $i \in \{1, 2, \dots, n\}$, and $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$
- There are $N = \prod_{i=1}^{n} l_i$ possible realizations of the network. The probability for each model $\sum_\gamma$ to be active is

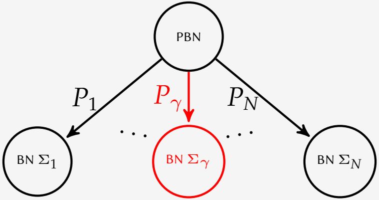
$$P_\gamma = Pr\{\text{network } \Sigma_\gamma \text{ is selected}\} = Pr\{\underbrace{f_1 = f_1^{\gamma_1}, \dots, f_n = f_n^{\gamma_n}}_{\text{independent}}\} = \prod_{i=1}^{n} p_i^{\gamma_i}$$

[2] D. Cheng, H. Qi, and Z. Li (2010). *Analysis and Control of Boolean Networks: A Semi-tensor Product Approach*. Communications and Control Engineering. Springer London.

PBNs as switched systems

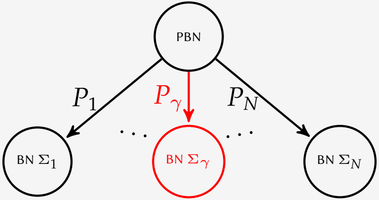


PBNs as switched systems

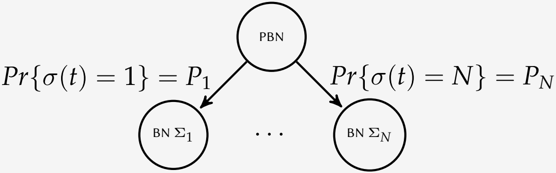


- At each t a model Σ_γ is selected, and the next state of the PBN is determined according to the corresponding BN

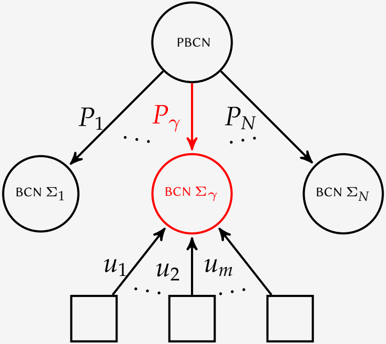
PBNs as switched systems



- At each t a model Σ_γ is selected, and the next state of the PBN is determined according to the corresponding BN
- PBN → switching systems, with $\sigma(t)$ taking values in $\{1, 2, \dots, \gamma, \dots, N\}$, and $Pr\{\sigma(t) = \gamma\} = P_\gamma$



Probabilistic Boolean control network



- A probabilistic Boolean control network (PBCN) is a combination of PBN and Boolean control network (BCN)

Logical form of PBCN

- The evolution of a PBCN in logical form is

$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \\ \qquad\qquad\vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \end{cases}$$

Logical form of PBCN

- The evolution of a PBCN in logical form is

$$\begin{cases} \mathcal{X}_1(t+1) = f_1\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \\ \mathcal{X}_2(t+1) = f_2\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \\ \qquad\qquad \vdots \\ \mathcal{X}_n(t+1) = f_n\big(\mathcal{X}_1(t), \mathcal{X}_2(t), \ldots, \mathcal{X}_n(t); \mu_1(t), \mu_2(t), \ldots, \mu_m(t)\big) \end{cases}$$

where

- $\mathcal{X}(t) \in \mathcal{B}^n, \mu(t) \in \mathcal{B}^m$
- at each time-step $t, f_i \in \{f_i^1, f_i^2, \ldots, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \times \mathcal{B}^m \to \mathcal{B}$, and with $Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, $\gamma_i \in \{1, 2, \ldots, l_i\}$, $i \in \{1, 2, \ldots, n\}$, and $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$
- Σ_γ models, with $\gamma = 1, 2, \ldots, N$, and $N = \prod_{i=1}^{n} l_i$:

$$P_\gamma = Pr\{\text{network} \sum_\gamma \text{is selected}\} = \prod_{i=1}^{n} p_i^{\gamma_i}$$

Algebraic form of PBCN

- PBCN can be converted into its algebraic form using semi-tensor product (STP):

$$\begin{cases} x_1(t+1) = f_1\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) = f_2\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ \qquad\qquad \vdots \\ x_n(t+1) = f_n\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_n} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \end{cases}$$

$\mathbf{x}(t) = \ltimes_{i=1}^{n} x_i(t) \in \mathcal{L}^{2^n}$, $\mathbf{u}(t) = \ltimes_{j=1}^{m} u_j(t) \in \mathcal{L}^{2^m}$, $M_{f_i} \in \mathcal{L}^{2^n \times 2^{n+m}}$, and $M_{f_i} \in \{M_{f_i}^1, M_{f_i}^2, \ldots, M_{f_i}^{l_i}\}$

Algebraic form of PBCN

- PBCN can be converted into its algebraic form using semi-tensor product (STP):

$$\begin{cases} x_1(t+1) = f_1\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) = f_2\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ \qquad\qquad \vdots \\ x_n(t+1) = f_n\big(x_1(t), x_2(t), \ldots, x_n(t); u_1(t), u_2(t), \ldots, u_m(t)\big) = \\ \qquad\qquad = M_{f_n} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \end{cases}$$

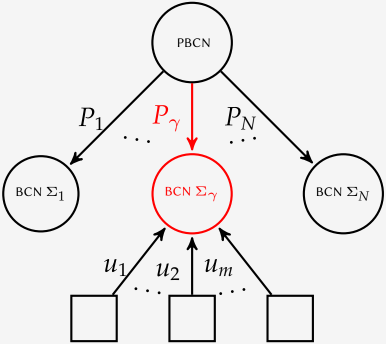
$\mathbf{x}(t) = \ltimes_{i=1}^{n} x_i(t) \in \mathcal{L}^{2^n}$, $\mathbf{u}(t) = \ltimes_{j=1}^{m} u_j(t) \in \mathcal{L}^{2^m}$, $M_{f_i} \in \mathcal{L}^{2^n \times 2^{n+m}}$, and $M_{f_i} \in \{M_{f_i}^1, M_{f_i}^2, \ldots, M_{f_i}^{l_i}\}$

- PBCN can be rewritten as:

$$\mathbf{x}(t+1) = L_{\gamma(t)} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t),$$

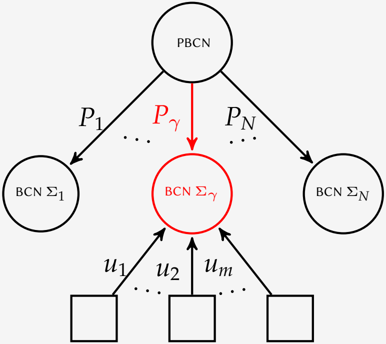
where $L_{\gamma(t)} \in \mathcal{L}^{2^n \times 2^{n+m}}$ is one of the N structure matrices

PBCN as switched system



- As a switching system, a PBCN can be written as
$$\mathbf{x}(t+1) = L \ltimes \gamma(t) \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t),$$
where $L = [L_1 \quad L_2 \quad \dots \quad L_N]$, $\gamma(t) \in \mathcal{L}^N$ selects with $Pr\{\gamma(t) = \delta_N^\gamma\} = P_\gamma$ the active structure matrix $L_\gamma \in \mathcal{L}^{2^n \times 2^{n+m}}$ taking values in $\{1, 2, \dots, N\}$, and $\mathbf{u}(t) \in \mathcal{L}^{2^m}$ is the input selecting the sub-model of L_γ

PBCN as switched system



- As a switching system, a PBCN can be written as
$$\mathbf{x}(t+1) = L \ltimes \gamma(t) \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t),$$
where $L = [L_1 \quad L_2 \quad \ldots \quad L_N]$, $\gamma(t) \in \mathcal{L}^N$ selects with $Pr\{\gamma(t) = \delta_N^\gamma\} = P_\gamma$ the active structure matrix $L_\gamma \in \mathcal{L}^{2^n \times 2^{n+m}}$ taking values in $\{1, 2, \ldots, N\}$, and $\mathbf{u}(t) \in \mathcal{L}^{2^m}$ is the input selecting the sub-model of L_γ
- Denote with $\mathbf{u}_{[0,T-1]} := \{\mathbf{u}(0), \ldots, \mathbf{u}(T-1)\}$, $T \in \mathbb{Z}_+$ a control sequence, and with $\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,T-1]})$ the state of a PBCN at time $t \in \mathbb{Z}_+$

Delving into PBCN

- PBCNs were first introduced by Shmulevich in 2002[3]

[3] I. Shmulevich et al. (Feb. 2002). "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks ". In: *Bioinformatics* 18.2, pp. 261–274.

Delving into PBCN

- PBCNs were first introduced by Shmulevich in 2002[3]
- The motivation for adopting PBCNs stems from several modeling advantages and applications:
    - Modeling complexity and uncertainty

[3] I. Shmulevich et al. (Feb. 2002). "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks ". In: *Bioinformatics* 18.2, pp. 261–274.

Delving into PBCN

- PBCNs were first introduced by Shmulevich in 2002[3]
- The motivation for adopting PBCNs stems from several modeling advantages and applications:
  - Modeling complexity and uncertainty
  - Developing therapeutic interventions

[3] I. Shmulevich et al. (Feb. 2002). "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks ". In: *Bioinformatics* 18.2, pp. 261–274.

Delving into PBCN

- PBCNs were first introduced by Shmulevich in 2002[3]
- The motivation for adopting PBCNs stems from several modeling advantages and applications:
  - Modeling complexity and uncertainty
  - Developing therapeutic interventions
  - Predictive modeling

[3] I. Shmulevich et al. (Feb. 2002). "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks ". In: *Bioinformatics* 18.2, pp. 261–274.

Delving into PBCN

- PBCNs were first introduced by Shmulevich in 2002[3]
- The motivation for adopting PBCNs stems from several modeling advantages and applications:
  - Modeling complexity and uncertainty
  - Developing therapeutic interventions
  - Predictive modeling
- The study of stabilization problems for PBCN is crucial for ensuring reliable and predictable behavior in complex biological and engineered systems

[3] I. Shmulevich et al. (Feb. 2002). "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks ". In: *Bioinformatics* 18.2, pp. 261–274.

Contents

Feedback stabilisation problem of PBCN[4]

- Control fixed point
  A state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is called a control fixed point if

$$\exists\ \mathbf{u}(t) \in \mathcal{L}^{2^m} : Pr\{\mathbf{x}(t+1; \mathbf{x}_e, \mathbf{u}(t)) = \mathbf{x}_e\} = 1$$

[4] X. Yang and H. Li (2022). "On state feedback asymptotical stabilization of probabilistic Boolean control networks". In: *Systems & Control Letters* 160, p. 105107.

Feedback stabilisation problem of PBCN[4]

- **Control fixed point**
  A state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is called a control fixed point if

$$\exists\ \mathbf{u}(t) \in \mathcal{L}^{2^m} : Pr\{\mathbf{x}(t+1; \mathbf{x}_e, \mathbf{u}(t)) = \mathbf{x}_e\} = 1$$

- **Stabilisation in distribution**
  Given $\mathbf{x}_e \in \mathcal{L}^{2^n}$, a PBCN is said to be stabilisable at $\mathbf{x}_e$ in distribution if

$$\exists\ \mathbf{u}_{[0,t-1]} : \lim_{t\to\infty} Pr\{\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,t]}) = \mathbf{x}_e\} = 1,\ \forall \mathbf{x}(0) \in \mathcal{L}^{2^n}$$

[4] X. Yang and H. Li (2022). "On state feedback asymptotical stabilization of probabilistic Boolean control networks". In: *Systems & Control Letters* 160, p. 105107.

Feedback stabilisation problem of PBCN[4]

- **Control fixed point**
  A state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is called a control fixed point if
  $$\exists\, \mathbf{u}(t) \in \mathcal{L}^{2^m} : Pr\{\mathbf{x}(t+1; \mathbf{x}_e, \mathbf{u}(t)) = \mathbf{x}_e\} = 1$$
- **Stabilisation in distribution**
  Given $\mathbf{x}_e \in \mathcal{L}^{2^n}$, a PBCN is said to be stabilisable at $\mathbf{x}_e$ in distribution if
  $$\exists\, \mathbf{u}_{[0,t-1]} : \lim_{t\to\infty} Pr\{\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,t]}) = \mathbf{x}_e\} = 1,\ \forall \mathbf{x}(0) \in \mathcal{L}^{2^n}$$
- **Stabilisation w.p.o.**
  Given $\mathbf{x}_e \in \mathcal{L}^{2^n}$, a PBCN is said to be finite-time stabilisable at $\mathbf{x}_e$ w.p.o. if
  $$\exists\, \mathbf{u}_{[0,T-1]} : Pr\{\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,T-1]}) = \mathbf{x}_e\} = 1,\ \forall t \geq T,\ \forall \mathbf{x}(0) \in \mathcal{L}^{2^n}$$

[4] X. Yang and H. Li (2022). “On state feedback asymptotical stabilization of probabilistic Boolean control networks”. In: *Systems & Control Letters* 160, p. 105107.

Feedback stabilisation problem of PBCN[4]

- **Control fixed point**
  A state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is called a control fixed point if
  $$\exists\ \mathbf{u}(t) \in \mathcal{L}^{2^m} : Pr\{\mathbf{x}(t+1; \mathbf{x}_e, \mathbf{u}(t)) = \mathbf{x}_e\} = 1$$
- **Stabilisation in distribution**
  Given $\mathbf{x}_e \in \mathcal{L}^{2^n}$, a PBCN is said to be stabilisable at $\mathbf{x}_e$ in distribution if
  $$\exists\ \mathbf{u}_{[0,t-1]} : \lim_{t\to\infty} Pr\{\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,t]}) = \mathbf{x}_e\} = 1,\ \forall \mathbf{x}(0) \in \mathcal{L}^{2^n}$$
- **Stabilisation w.p.o.**
  Given $\mathbf{x}_e \in \mathcal{L}^{2^n}$, a PBCN is said to be finite-time stabilisable at $\mathbf{x}_e$ w.p.o. if
  $$\exists\ \mathbf{u}_{[0,T-1]} : Pr\{\mathbf{x}(t; \mathbf{x}(0), \mathbf{u}_{[0,T-1]}) = \mathbf{x}_e\} = 1,\ \forall t \geq T,\ \forall \mathbf{x}(0) \in \mathcal{L}^{2^n}$$
- **Feedback stabilisation problem**
  Find a state feedback control law $\mathbf{u}(t) = \mathcal{K}(\mathbf{x}(t)), \forall \mathbf{x}(t) \in \mathcal{L}^{2^n}$ such that the PBCN is stabilised at $\mathbf{x}_e$

[4] X. Yang and H. Li (2022). "On state feedback asymptotical stabilization of probabilistic Boolean control networks". In: *Systems & Control Letters* 160, p. 105107.

State feedback control of PBCN[5]

- The problem of stabilisation can be solved using a time-invariant feedback law as:

$$\mathbf{u}(t) = K \ltimes \mathbf{x}(t),$$

where $K \in \mathcal{L}^{2^m \times 2^n}$ is the structure feedback matrix of the control law

[5] R. Li, M. Yang, and T. Chu (2014). "State feedback stabilization for probabilistic Boolean networks". In: *Automatica* 50.4, pp. 1272–1278.

State feedback control of PBCN[5]

- The problem of stabilisation can be solved using a time-invariant feedback law as:

$$\mathbf{u}(t) = K \ltimes \mathbf{x}(t),$$

where $K \in \mathcal{L}^{2^m \times 2^n}$ is the structure feedback matrix of the control law

- The overall closed-loop system

$$\mathbf{x}(t+1) = L_{in} \ltimes K \ltimes \mathbf{x}(t) \ltimes \mathbf{x}(t)$$

is stable w.p.o., where $L_{in} \in \mathcal{L}^{2^m \times 2^{n+m}}$ is defined as:

$$L_{in} = \sum_{\gamma=1}^{N} L_\gamma$$

[5] R. Li, M. Yang, and T. Chu (2014). "State feedback stabilization for probabilistic Boolean networks". In: *Automatica* 50.4, pp. 1272–1278.

State feedback control of PBCN

Recursive sets

Assume $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is the fixed point to which the network needs to be stabilised. Define $\{\Omega_k(\mathbf{x}_e)\}$ as

$$\begin{cases} \Omega_1(\mathbf{x}_e) = \{a \in \mathcal{L}^{2^n} : \text{ there is a } \mathbf{u} \in \mathcal{L}^{2^m} \text{ such that} \\ \qquad\qquad Pr\{\mathbf{x}(t+1) = \mathbf{x}_e | \mathbf{x}(t) = a, \mathbf{u}(t) = \mathbf{u}\} = 1\} \\ \Omega_{k+1}(\mathbf{x}_e) = \{a \in \mathcal{L}^{2^n} : \text{ there is a } \mathbf{u} \in \mathcal{L}^{2^m} \text{ such that the conditions} \\ \qquad\qquad Pr\{\mathbf{x}(t+1) = b | \mathbf{x}(t) = a, \mathbf{u}(t) = \mathbf{u}\} > 0, b \in \mathcal{L}^{2^n} \\ \qquad\qquad \text{imply that } b \in \{\Omega_k(\mathbf{x}_e)\}, k = 1, 2, \dots\} \end{cases}$$

State feedback control of PBCN

Recursive sets

Assume $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is the fixed point to which the network needs to be stabilised. Define $\{\Omega_k(\mathbf{x}_e)\}$ as

$$\begin{cases} \Omega_1(\mathbf{x}_e) = \{a \in \mathcal{L}^{2^n} : \text{ there is a } \mathbf{u} \in \mathcal{L}^{2^m} \text{ such that} \\ \qquad\qquad Pr\{\mathbf{x}(t+1) = \mathbf{x}_e | \mathbf{x}(t) = a, \mathbf{u}(t) = \mathbf{u}\} = 1\} \\ \Omega_{k+1}(\mathbf{x}_e) = \{a \in \mathcal{L}^{2^n} : \text{ there is a } \mathbf{u} \in \mathcal{L}^{2^m} \text{ such that the conditions} \\ \qquad\qquad Pr\{\mathbf{x}(t+1) = b | \mathbf{x}(t) = a, \mathbf{u}(t) = \mathbf{u}\} > 0, b \in \mathcal{L}^{2^n} \\ \qquad\qquad \text{imply that } b \in \{\Omega_k(\mathbf{x}_e)\}, k = 1, 2, \ldots\} \end{cases}$$

- **Remark.** The structure of $\Omega_k(\mathbf{x}_e)$ does not depend on the probabilities $P_1, P_2, \ldots, P_N$ that a certain BCN is selected. → Does not influence whether a PBCN can be stabilised by state feedback

State feedback control of PBCN

- **[Theorem]** Consider a PBCN, and let $\mathbf{x}_e = \ltimes_{i=1}^{n} x_{e_i}(t)$. If there is a state feedback control $\mathbf{u}(\cdot)$ such that the PBCN is sabilisable at $\mathbf{x}_e$ w.p.o., then it holds that
  - $\mathbf{x}_e \in \Omega_1(\mathbf{x}_e)$
  - There exists an integer $G \leq 2^n - 1$ such that $\Omega_G(\mathbf{x}_e) = \mathcal{L}^{2^n}$

[6] R. Li, M. Yang, and T. Chu (2014). “State feedback stabilization for probabilistic Boolean networks”. In: *Automatica* 50.4, pp. 1272–1278.

State feedback control of PBCN

- **[Theorem]** Consider a PBCN, and let $\mathbf{x}_e = \ltimes_{i=1}^{n} x_{e_i}(t)$. If there is a state feedback control $\mathbf{u}(\cdot)$ such that the PBCN is sabilisable at $\mathbf{x}_e$ w.p.o., then it holds that
  - $\mathbf{x}_e \in \Omega_1(\mathbf{x}_e)$
  - There exists an integer $G \leq 2^n - 1$ such that $\Omega_G(\mathbf{x}_e) = \mathcal{L}^{2^n}$
- Li et al.[6] defined a constructive method to obtain v_i, $i = 1, \dots, 2^n$ such that, when the structure matrix K of the feedback law has the following shape

$$K = \begin{bmatrix} v_1 & v_2 & \cdots & v_{2^n} \end{bmatrix},$$

the PBCN is stabilisable at $\mathbf{x}_e$ w.p.o.

[6] R. Li, M. Yang, and T. Chu (2014). “State feedback stabilization for probabilistic Boolean networks”. In: *Automatica* 50.4, pp. 1272–1278.

State feedback control of PBCN

- **[Theorem]** Consider a PBCN, and let $\mathbf{x}_e = \ltimes_{i=1}^{n} x_{e_i}(t)$. If there is a state feedback control $\mathbf{u}(\cdot)$ such that the PBCN is sabilisable at $\mathbf{x}_e$ w.p.o., then it holds that
  - $\mathbf{x}_e \in \Omega_1(\mathbf{x}_e)$
  - There exists an integer $G \leq 2^n - 1$ such that $\Omega_G(\mathbf{x}_e) = \mathcal{L}^{2^n}$
- Li et al.[6] defined a constructive method to obtain v_i, $i = 1, \ldots, 2^n$ such that, when the structure matrix K of the feedback law has the following shape

$$K = [v_1 \quad v_2 \quad \cdots \quad v_{2^n}],$$

  the PBCN is stabilisable at $\mathbf{x}_e$ w.p.o.
- **Remark.** If a PBCN can be stabilised at $\mathbf{x}_e$ w.p.o., then every $\mathbf{x}$ can be steered to $\mathbf{x}_e$ w.p.o. using random control.

[6] R. Li, M. Yang, and T. Chu (2014). "State feedback stabilization for probabilistic Boolean networks". In: *Automatica* 50.4, pp. 1272–1278.

State feedback control of PBCN

- **[Theorem]** Consider a PBCN, and let $\mathbf{x}_e = \ltimes_{i=1}^{n} x_{e_i}(t)$. If there is a state feedback control $\mathbf{u}(\cdot)$ such that the PBCN is sabilisable at $\mathbf{x}_e$ w.p.o., then it holds that
  - $\mathbf{x}_e \in \Omega_1(\mathbf{x}_e)$
  - There exists an integer $G \leq 2^n - 1$ such that $\Omega_G(\mathbf{x}_e) = \mathcal{L}^{2^n}$
- Li et al.[6] defined a constructive method to obtain v_i, $i = 1, \ldots, 2^n$ such that, when the structure matrix K of the feedback law has the following shape

$$K = [v_1 \quad v_2 \quad \cdots \quad v_{2^n}],$$

  the PBCN is stabilisable at $\mathbf{x}_e$ w.p.o.
- **Remark.** If a PBCN can be stabilised at $\mathbf{x}_e$ w.p.o., then every $\mathbf{x}$ can be steered to $\mathbf{x}_e$ w.p.o. using random control. However, the above constructive method and the theorem ensure the **shortest path to stabilise the network**

[6] R. Li, M. Yang, and T. Chu (2014). "State feedback stabilization for probabilistic Boolean networks". In: *Automatica* 50.4, pp. 1272–1278.

Model-based methods for PBCN stabilisation

- Some modifications to PBCN stabilisation problem:
  - Y. Liu et al.[7] built a state feedback controller to reach a target state while avoiding undesirable states
  - X. Yang et al.[8] proposed a computationally efficient solution for the asymptotic stabilisation of PBCN
  - A. Yerudkar et al.[9] developed the solution of the output tracking model and showed that for a constant reference signal the problem can be cast to a state feedback stabilisation problem

[7] Y. Liu et al. (Feb. 2015). "Controllability of Probabilistic Boolean Control Networks Based on Transition Probability Matrices". In: *Automatica* 52.C, pp. 340–345.

[8] X. Yang and H. Li (2022). "On state feedback asymptotical stabilization of probabilistic Boolean control networks". In: *Systems & Control Letters* 160, p. 105107.

[9] A. Yerudkar, C. Del Vecchio, and L. Glielmo (2019). "Output Tracking Control of Probabilistic Boolean Control Networks". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*.

Model-based methods for PBCN stabilisation

- Some modifications to PBCN stabilisation problem:
  - Y. Liu et al.[7] built a state feedback controller to reach a target state while avoiding undesirable states
  - X. Yang et al.[8] proposed a computationally efficient solution for the asymptotic stabilisation of PBCN
  - A. Yerudkar et al.[9] developed the solution of the output tracking model and showed that for a constant reference signal the problem can be cast to a state feedback stabilisation problem

✗ Require a full knowledge of the underlying dynamics of the system

✗ The humongous nature of biological systems proves to be a bottleneck of the modeling part and hinders the applicability of model-based methods

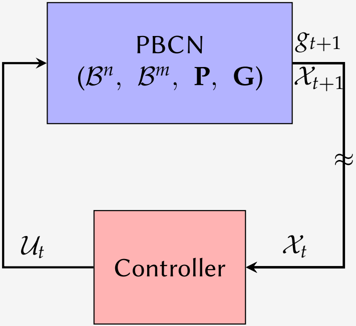
✗ Model-based methods are limited by the computational complexity

[7] Y. Liu et al. (Feb. 2015). "Controllability of Probabilistic Boolean Control Networks Based on Transition Probability Matrices". In: *Automatica* 52.C, pp. 340–345.

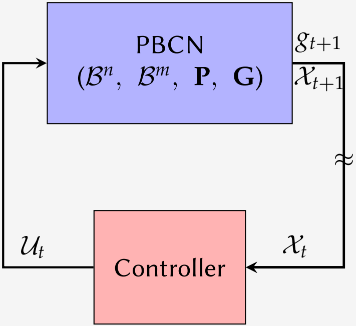
[8] X. Yang and H. Li (2022). "On state feedback asymptotical stabilization of probabilistic Boolean control networks". In: *Systems & Control Letters* 160, p. 105107.

[9] A. Yerudkar, C. Del Vecchio, and L. Glielmo (2019). "Output Tracking Control of Probabilistic Boolean Control Networks". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*.

Model-free state feedback control: a reinforcement learning (RL) approach

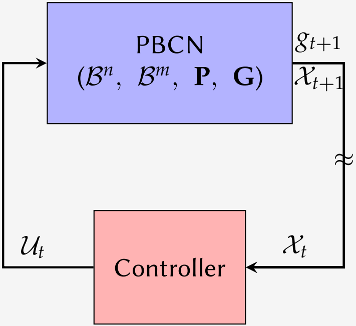


Model-free state feedback control: a reinforcement learning (RL) approach



- $\mathcal{B}^n$ is the set of states
- $\mathcal{B}^m$ is the set of actions
- $\mathbf{P} : \mathcal{B}^n \times \mathcal{B}^m \times \mathcal{B}^n \to [0,1]$ is the state transition probability distribution, with $\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}_{t+1}} = \mathrm{P}\{\mathcal{X}_{t+1}|\mathcal{X}_t,\mathcal{U}_t\}$
- $\mathbf{G} : \mathcal{B}^n \times \mathcal{B}^m \times \mathcal{B}^n \to \mathbb{R}$ is the cost function, with $\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}_{t+1}} = \mathbb{E}[g_{t+1}|\mathcal{X}_t,\mathcal{U}_t]$

Model-free state feedback control: a reinforcement learning (RL) approach

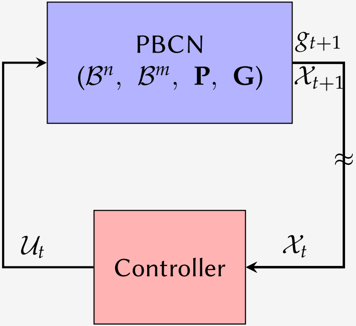


- $\mathcal{B}^n$ is the set of states
- $\mathcal{B}^m$ is the set of actions
- $\mathbf{P} : \mathcal{B}^n \times \mathcal{B}^m \times \mathcal{B}^n \to [0,1]$ is the state transition probability distribution, with $\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}_{t+1}} = \mathrm{P}\{\mathcal{X}_{t+1}|\mathcal{X}_t,\mathcal{U}_t\}$
- $\mathbf{G} : \mathcal{B}^n \times \mathcal{B}^m \times \mathcal{B}^n \to \mathbb{R}$ is the cost function, with $\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}_{t+1}} = \mathbb{E}[g_{t+1}|\mathcal{X}_t,\mathcal{U}_t]$

Generated trajectory

$\mathcal{X}_0,\ \mathcal{U}_0,\ \mathcal{X}_1,\ g_1,\ \mathcal{U}_1,\ \mathcal{X}_2,\ g_2,\ \mathcal{U}_2, \ldots$

Model-free state feedback control: an RL approach



RL objective

Find a policy $\pi : \mathcal{B}^n \times \mathcal{B}^m \to [0,1]$ such that

$$\min_{\pi} \mathbb{E}_{\pi,\mathbf{P}} \left[\sum_{t=0}^{\infty} \gamma^t g_{t+1} \right], \forall \mathcal{X}_0 \in \mathcal{B}^n,$$

where $\gamma \in [0,1)$ is the discount factor

Generated trajectory

$\mathcal{X}_0,\ \mathcal{U}_0,\ \mathcal{X}_1,\ g_1,\ \mathcal{U}_1,\ \mathcal{X}_2,\ g_2,\ \mathcal{U}_2, \ldots$

Dynamic programming (DP) solutions

- The policy can be learned indirectly using the value function

$$v^{\pi}(\mathcal{X}_0) := \mathbb{E}_{\pi,\mathbf{P}}\left[\sum_{i=0}^{\infty}\gamma^i g_{i+1}\middle|\mathcal{X}_0\right], \text{ for all } \mathcal{X}_0 \in \mathcal{B}^n,$$

and the action-value function

$$q^{\pi}(\mathcal{X}_0,\mathcal{U}_0) := \mathbb{E}_{\pi,\mathbf{P}}\left[\sum_{i=0}^{\infty}\gamma^i g_{i+1}\middle|\mathcal{X}_0,\mathcal{U}_0\right], \text{ for all } \mathcal{X}_0 \in \mathcal{B}^n$$

- Recursive Bellman equation

$$q^{\pi}(\mathcal{X}_t,\mathcal{U}_t) = \sum_{\mathcal{X}\in\mathcal{B}^n}\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}}\left[\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}} + \gamma\sum_{\mathcal{U}\in\mathcal{B}^m}\pi(\mathcal{U}|\mathcal{X})q^{\pi}(\mathcal{X},\mathcal{U})\right]$$

- Bellman optimality equation

$$q^{*}(\mathcal{X}_t,\mathcal{U}_t) := q^{\pi^*}(\mathcal{X}_t,\mathcal{U}_t) = \sum_{\mathcal{X}\in\mathcal{B}^n}\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}}\left[\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}} + \gamma\min_{\mathcal{U}}q^{*}(\mathcal{X},\mathcal{U})\right]$$

Dynamic programming (DP) solutions

- The policy can be learned indirectly using the value function

$$v^{\pi}(\mathcal{X}_0) := \mathbb{E}_{\pi,\mathbf{P}}\left[\sum_{i=0}^{\infty}\gamma^i g_{i+1}\middle|\mathcal{X}_0\right],\ \text{for all } \mathcal{X}_0 \in \mathcal{B}^n,$$

and the action-value function

$$q^{\pi}(\mathcal{X}_0,\mathcal{U}_0) := \mathbb{E}_{\pi,\mathbf{P}}\left[\sum_{i=0}^{\infty}\gamma^i g_{i+1}\middle|\mathcal{X}_0,\mathcal{U}_0\right],\ \text{for all } \mathcal{X}_0 \in \mathcal{B}^n$$

- Recursive Bellman equation

$$q^{\pi}(\mathcal{X}_t,\mathcal{U}_t) = \sum_{\mathcal{X}\in\mathcal{B}^n}\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}}\left[\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}} + \gamma\sum_{\mathcal{U}\in\mathcal{B}^m}\pi(\mathcal{U}|\mathcal{X})q^{\pi}(\mathcal{X},\mathcal{U})\right]$$

- Bellman optimality equation

$$q^{*}(\mathcal{X}_t,\mathcal{U}_t) := q^{\pi^*}(\mathcal{X}_t,\mathcal{U}_t) = \sum_{\mathcal{X}\in\mathcal{B}^n}\mathbf{P}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}}[\mathbf{G}^{\mathcal{U}_t}_{\mathcal{X}_t,\mathcal{X}} + \gamma\min_{\mathcal{U}} q^{*}(\mathcal{X},\mathcal{U})]$$

Exact DP requires knowledge of the model!

The “tabular” model-free approach: Q-Learning (QL)

- QL algorithm solves the recursive Bellman equation iteratively using samples
- It updates estimates of $Q(\cdot,\cdot)$ based on other learned estimates, with the following update rule

$$Q_{t+1}(\mathcal{X}_t,\mathcal{U}_t) = Q_t(\mathcal{X}_t,\mathcal{U}_t) + \alpha_t[g_{t+1} + \gamma \min_{\mathcal{U}\in\mathcal{B}^m} Q_t(\mathcal{X}_{t+1},\mathcal{U}) - Q_t(\mathcal{X}_t,\mathcal{U}_t)]$$

States \ *Actions*	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\dots$
$\mathcal{X}_0$	$Q(\mathcal{X}_0,\mathcal{U}_0)$	$Q(\mathcal{X}_0,\mathcal{U}_1)$	$Q(\mathcal{X}_0,\mathcal{U}_2)$	$\dots$
$\mathcal{X}_1$	$Q(\mathcal{X}_1,\mathcal{U}_0)$	$Q(\mathcal{X}_1,\mathcal{U}_1)$	$Q(\mathcal{X}_1,\mathcal{U}_2)$	$\dots$
$\mathcal{X}_2$	$Q(\mathcal{X}_2,\mathcal{U}_0)$	$Q(\mathcal{X}_2,\mathcal{U}_1)$	$Q(\mathcal{X}_2,\mathcal{U}_2)$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

[10] C. J. Watkins and P. Dayan (1992). “Q-learning”. In: *Machine learning* 8.3-4, pp. 279–292.

The “tabular” model-free approach: Q-Learning (QL)

- QL algorithm solves the recursive Bellman equation iteratively using samples
- It updates estimates of $Q(\cdot,\cdot)$ based on other learned estimates, with the following update rule

$$Q_{t+1}(\mathcal{X}_t,\mathcal{U}_t) = Q_t(\mathcal{X}_t,\mathcal{U}_t) + \alpha_t[g_{t+1} + \gamma \min_{\mathcal{U}\in\mathcal{B}^m} Q_t(\mathcal{X}_{t+1},\mathcal{U}) - Q_t(\mathcal{X}_t,\mathcal{U}_t)]$$

States \ *Actions*	$\mathcal{U}_0$	$\mathcal{U}_1$	$\mathcal{U}_2$	$\dots$
$\mathcal{X}_0$	$Q(\mathcal{X}_0,\mathcal{U}_0)$	$Q(\mathcal{X}_0,\mathcal{U}_1)$	$Q(\mathcal{X}_0,\mathcal{U}_2)$	$\dots$
$\mathcal{X}_1$	$Q(\mathcal{X}_1,\mathcal{U}_0)$	$Q(\mathcal{X}_1,\mathcal{U}_1)$	$Q(\mathcal{X}_1,\mathcal{U}_2)$	$\dots$
$\mathcal{X}_2$	$Q(\mathcal{X}_2,\mathcal{U}_0)$	$Q(\mathcal{X}_2,\mathcal{U}_1)$	$Q(\mathcal{X}_2,\mathcal{U}_2)$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

✓ QL has been proven to converge at least asymptotically to $Q^*(\cdot,\cdot)$ under standard stochastic approximation conditions[10]. The optimal policy can be obtained as

$$\pi^*(\mathcal{X}) = \arg\min_{\mathcal{U}\in\mathcal{B}^m} Q^*(\mathcal{X},\mathcal{U}), \quad \forall \mathcal{X} \in \mathcal{B}^n$$

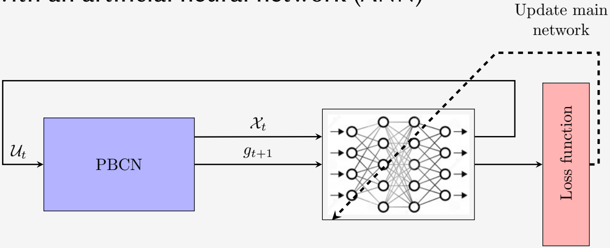
✓ It is a simple yet powerful algorithm

✗ The tabular architecture only applies to relatively small state-action spaces

[10] C. J. Watkins and P. Dayan (1992). “Q-learning”. In: *Machine learning* 8.3-4, pp. 279–292.

Non-tabular RL approach: deep Q network (DQN)

- Given a state $\mathcal{X}_t$, the action-value function is estimated using function approximators, e.g., with an artificial neural network (ANN)



- The ANN is trained to minimize a differentiable loss function, namely the Bellman error

$$\mathcal{E}(\mathbf{W}) = \frac{1}{2}\left|\left|Q(\mathcal{X}_t, \mathcal{U}_t, \mathbf{W}) - \mathcal{Y}_{t+1}\right|\right|^2,$$

where $\mathcal{Y}_{t+1} = g_{t+1} + \gamma \min_{\mathcal{U}\in\mathcal{B}^m} Q(\mathcal{X}_{t+1}, \mathcal{U}, \mathbf{W})$ is the target value

- Then, $\mathbf{W}$ can be updated through stochastic gradient descent (SGD) method

$$\mathbf{W} = \mathbf{W} - \alpha\nabla_{\mathbf{W}}\mathcal{E}(\mathbf{W}) = \mathbf{W} - \alpha\left[Q(\mathcal{X}_t, \mathcal{U}_t, \mathbf{W}) - \mathcal{Y}_{t+1}\right]\nabla_{\mathbf{W}} Q(\mathcal{X}_t, \mathcal{U}_t, \mathbf{W})$$

DQN limitations

- ✗ Sequential states are strongly correlated, and SGD method assumes that samples are uncorrelated
- ✗ Target value depends on the ANN parameters $\mathbf{W}$, and consequently its value changes over time-steps

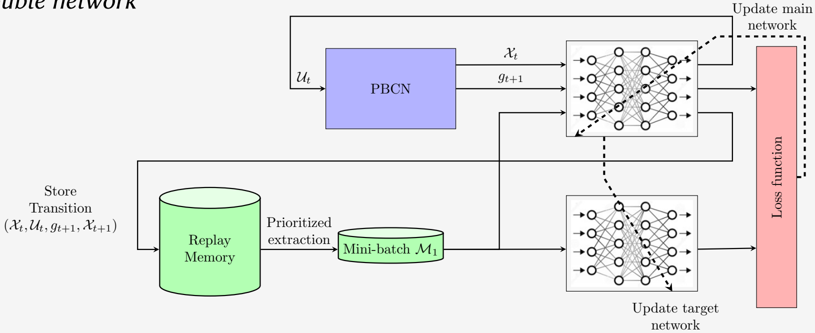
$$\mathcal{E}(\mathbf{W}) = \frac{1}{2}\left|\left|Q(\mathcal{X}_t, \mathcal{U}_t, \mathbf{W}) - \left(g_{t+1} + \gamma \min_{\mathcal{U}\in\mathcal{B}^m} Q(\mathcal{X}_{t+1}, \mathcal{U}, \mathbf{W})\right)\right|\right|^2$$

$$\Downarrow$$

High variance

Double deep Q network with prioritized experience replay (DDQN + PER)[11]

To limit stability issues, DDQN + PER introduces the *prioritized experience replay*, and a *double network*



✗ There are still no convergence guarantees

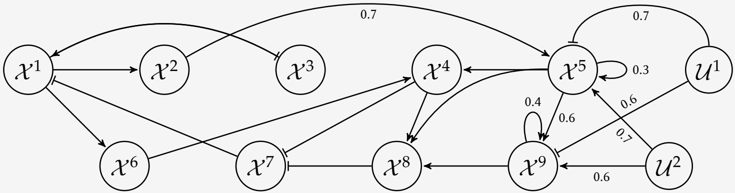
✓ Very effective in practice

[11] H. van Hasselt, A. Guez, and D. Silver (2016). "Deep Reinforcement Learning with Double Q-learning". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Contents

Example: finite-time stabilization of PBCN using QL[12]

- Lactose operon in Escherichia Coli (9 genes, 2 input genes)

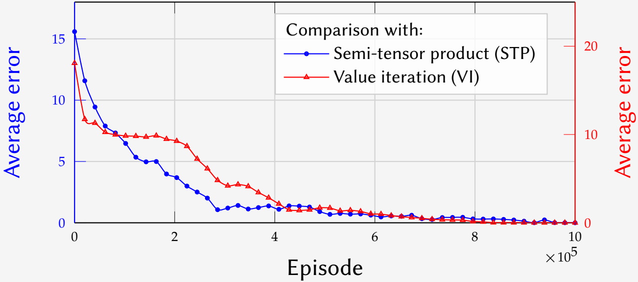


- Equilibrium state: $\bar{\mathcal{X}}_e = (1,1,1,1,1,1,0,1,1) \equiv$ the lactose operon attains the ON state

- Cost function: $$g_{t+1} = \begin{cases} -1 & \text{if } \mathcal{X}_{t+1} = \bar{\mathcal{X}}_e \\ 1 & \text{if } (\mathcal{X}_{t+1} \neq \bar{\mathcal{X}}_e) \wedge (\mathcal{X}_{t+1} = \mathcal{X}_t) \\ 0 & \text{otherwise} \end{cases}$$

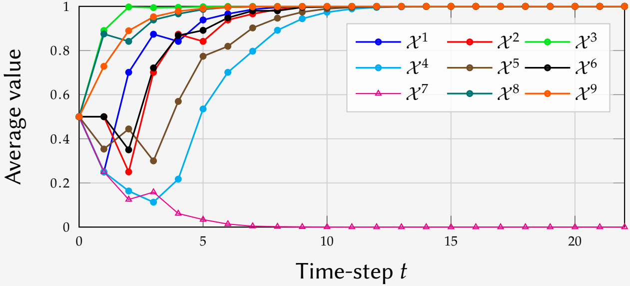
[12] A. Acernese, A. Yerudkar, L. Glielmo, and C. Del Vecchio (Jan. 2021). “Reinforcement Learning Approach to Feedback Stabilization Problem of Probabilistic Boolean Control Networks”. In: *IEEE Control Systems Letters* 5.1, pp. 337–342.

Comparison with model-based methods



- Performance evaluated as the average error (over M episodes) between the optimal value function v_{VI}^* and the current estimate of the action-value function (in VI case), and between the optimal state feedback law and the current estimated policy (in STP case)
- As the error tends to zero towards the end of the training, QL approaches an optimal solution

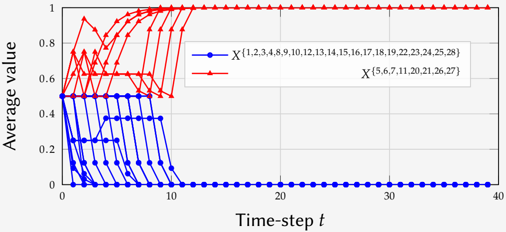
Performance of the proposed algorithm

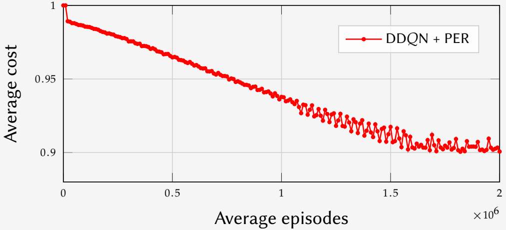


- $\bar{\mathcal{X}}_e = (1, 1, 1, 1, 1, 1, 0, 1, 1)$.
- Average evolution of genes over 5.12×10^5 episodes.
- The agent is able to stabilize the lactose operon system at $\bar{\mathcal{X}}_e$ w.p.o. in at most 23 steps.

Example: constant reference tracking of a PBCN[13]

- 28-genes, 3-inputs, with $|\mathcal{B}^n| \times |\mathcal{B}^m| \cong 2 \times 10^9$
- $\mathcal{X}_{r_t} : \bar{\mathcal{B}}^n \to (0,0,0,0,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0)$





[13] A. Acernese, A. Yerudkar, L. Glielmo, and C. Del Vecchio (2020). "Double Deep-Q Learning-Based Output Tracking of Probabilistic Boolean Control Networks". In: *IEEE Access* 8, pp. 199254–199265.

Self-triggered control (STC) co-design for PBCNs

The STC co-design of PBCNs consists in collectively computing the feedback control action to take and the next time instant to update the action, being in the current state

$$\mathbf{u}(t) = \mathcal{K}(\mathbf{x}(t_k)), \quad t \in [t_k,\ t_{k+1}),\ k \in \mathbb{Z}_+,$$
$$t_{k+1} = t_k + \tau(\mathbf{x}(t_k))^a$$

[a] $\mathbf{u}$, $\mathbf{x}$, are in canonical vector form

- Unlike the conventional state feedback control that is updated at each time-step, the STC follows a self-triggering schedule to provide an optimal control law when necessary
- *Resource-aware control*: perform actions when needed and share resources considering the limited availability

Model-free STC co-design with QL

- Define the macro-action (MA) $\mathsf{U}_t := \big(\mu_{act}(\mathcal{X}_t),\ \mu_{com}(\mathcal{X}_t)\big)$, where $\mu_{act} : \mathcal{B}^n \to \mathcal{B}^m$, and $\mu_{com} : \mathcal{B}^n \to \mathbb{Z}_+$

[14] A. Acernese* et al. (Nov. 2021). "Model-Free Self-Triggered Control Co-Design for Probabilistic Boolean Control Networks". In: *IEEE Control Systems Letters* 5.5, pp. 1639–1644.

Model-free STC co-design with QL

- Define the macro-action (MA) $\mathsf{U}_t := \big(\mu_{act}(\mathcal{X}_t),\ \mu_{com}(\mathcal{X}_t)\big)$, where $\mu_{act} : \mathcal{B}^n \to \mathcal{B}^m$, and $\mu_{com} : \mathcal{B}^n \to \mathbb{Z}_+$
- Define multi-time models for MAs, that predict expected states and expected costs after performing MAs, appropriately discounted
- Bellman's equations also hold on MAs with multi-time models:

$$q_\pi(\mathcal{X}_{t_k}, \mathsf{U}_{t_k}) = \mathbf{G}^{\mathsf{U}_{t_k}}_{\mathcal{X}_{t_k},\mathcal{X}_{t_{k+1}}} + \sum_{\mathcal{X}_{t_{k+1}}} \mathcal{P}^{\mathsf{U}_{t_k}}_{\mathcal{X}_{t_k},\mathcal{X}_{t_{k+1}}} \sum_{\mathsf{U}\in\mathcal{B}^{m+}} \pi(\mathcal{X}_{t_{k+1}}, \mathsf{U}) q_\pi(\mathcal{X}_{t_{k+1}}, \mathsf{U})$$

[14] A. Acernese* et al. (Nov. 2021). "Model-Free Self-Triggered Control Co-Design for Probabilistic Boolean Control Networks". In: *IEEE Control Systems Letters* 5.5, pp. 1639–1644.

Model-free STC co-design with QL

- Define the macro-action (MA) $\mathsf{U}_t := \big(\mu_{act}(\mathcal{X}_t),\ \mu_{com}(\mathcal{X}_t)\big)$, where $\mu_{act} : \mathcal{B}^n \to \mathcal{B}^m$, and $\mu_{com} : \mathcal{B}^n \to \mathbb{Z}_+$
- Define multi-time models for MAs, that predict expected states and expected costs after performing MAs, appropriately discounted
- Bellman's equations also hold on MAs with multi-time models:

$$q_\pi(\mathcal{X}_{t_k}, \mathsf{U}_{t_k}) = \mathbf{G}^{\mathsf{U}_{t_k}}_{\mathcal{X}_{t_k},\mathcal{X}_{t_{k+1}}} + \sum_{\mathcal{X}_{t_{k+1}}} \mathcal{P}^{\mathsf{U}_{t_k}}_{\mathcal{X}_{t_k},\mathcal{X}_{t_{k+1}}} \sum_{\mathsf{U}\in\mathcal{B}^{m+}} \pi(\mathcal{X}_{t_{k+1}}, \mathsf{U}) q_\pi(\mathcal{X}_{t_{k+1}}, \mathsf{U})$$

- In a model-free framework, we defined the self-triggered Q learning (STQL)[14] algorithm (proven to converge):

$$\begin{aligned} Q_{t_{k+1}}(\mathcal{X}_{t_k}, \mathsf{U}_{t_k}) = & Q_{t_k}(\mathcal{X}_{t_k}, \mathsf{U}_{t_k}) + \alpha_{t_k} \Bigg[\sum_{i=1}^{\tau(\mathcal{X}_{t_k})} \gamma^{i-1} g_{t_k+i} \\ & + \gamma^{\tau(\mathcal{X}_{t_k})} \min_{\mathsf{U}\in\mathcal{B}^{m+}} Q_{t_k}(\mathcal{X}_{t_{k+1}}, \mathsf{U}) - Q_{t_k}(\mathcal{X}_{t_k}, \mathsf{U}_{t_k}) \Bigg] \end{aligned}$$

[14] A. Acernese* et al. (Nov. 2021). "Model-Free Self-Triggered Control Co-Design for Probabilistic Boolean Control Networks". In: *IEEE Control Systems Letters* 5.5, pp. 1639–1644.

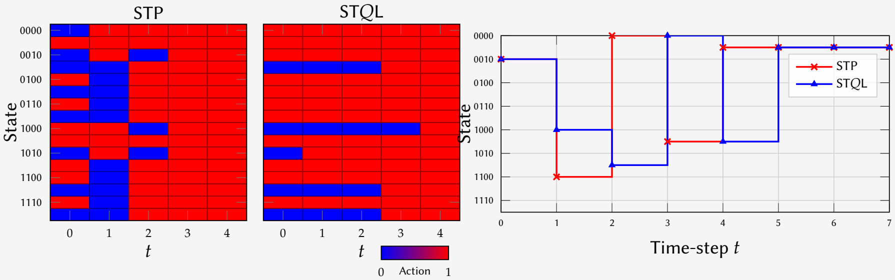
Example: STQL for stabilization of GRNs

- We consider a 4-gene PBCN model of the bacteriophage λ, a virus which can infect Escherichia coli bacteria:

$$\mathcal{X}^1_+ = \neg\mathcal{X}^2 \wedge \neg\mathcal{X}^4 \qquad \mathcal{X}^3_+ = \begin{cases} \neg\mathcal{X}^2 \wedge \neg\mathcal{X}^4 \wedge \mathcal{X}^1, & \mathrm{P} = 0.7 \\ 0, & \mathrm{P} = 0.3 \end{cases}$$

$$\mathcal{X}^2_+ = \neg\mathcal{X}^4 \wedge \neg\mathcal{U} \wedge (\mathcal{X}^2 \vee \mathcal{X}^3) \qquad \mathcal{X}^4_+ = \neg\mathcal{X}^2 \wedge \neg\mathcal{X}^3$$

- Equilibrium state: $\bar{\mathcal{X}}_e = (0, 0, 0, 1)$



- STQL algorithm stabilizes the system at $\bar{\mathcal{X}}_e$ while minimizing the action changes

Contents

Conclusions and perspectives

- The stabilization problem of PBCNs has been addressed

Conclusions and perspectives

- The stabilization problem of PBCNs has been addressed
- Model-based solutions for deriving feedback stabilization laws have been discussed, and, in light of their limitations, model-free solutions have been developed

Conclusions and perspectives

- The stabilization problem of PBCNs has been addressed
- Model-based solutions for deriving feedback stabilization laws have been discussed, and, in light of their limitations, model-free solutions have been developed
- The application of model-free method to various gene regulatory networks has been proposed

Conclusions and perspectives

- The stabilization problem of PBCNs has been addressed
- Model-based solutions for deriving feedback stabilization laws have been discussed, and, in light of their limitations, model-free solutions have been developed
- The application of model-free method to various gene regulatory networks has been proposed
- Emerging trends in PBCN control include pinning control and the adaptation of model-based control tools. Broadening the range of PBCN applications represents a promising direction for future research