Model *Learnable* Petri Net dalam struktur Neural Network berbasis Aljabar Max-Plus

Teosofi Hidayah Agung – 5002221132

Departemen Matematika
Institut Teknologi Sepuluh Nopember

Senin, 2 Juni 2025

Daftar Isi

Latar Belakang

- **Petri Net**: Model formal untuk sistem diskrit yang interpretable
- **Masalah**: Petri net tidak dirancang untuk pembelajaran adaptif
- **Solusi**: Menghubungkan Petri net dengan neural network melalui *max-plus algebra*

Tujuan

Membuat representasi Petri net yang dapat dipelajari menggunakan algoritma pembelajaran mesin standar, sambil mempertahankan interpretabilitas model.

Motivasi

Kenapa penting?

1. Production scheduling dengan waktu proses dinamis
2. Sistem manufaktur dengan ketidakpastian timing
3. Perlu model yang:
    - Interpretable (dapat dijelaskan)
    - Learnable (dapat dipelajari dari data)
    - Scalable (dapat ditingkatkan)

Definisi 2.1 (Petri Net)

Petri net adalah pasangan $(\mathcal{G}, \mu_0)$ di mana:

- $\mathcal{G} = (P, T, A)$: graf bipartit terarah
- $P = \{p_1, \ldots, p_m\}$: himpunan *places*
- $T = \{t_1, \ldots, t_n\}$: himpunan *transitions*
- $A \subseteq (P \times T) \cup (T \times P)$: himpunan *arcs*
- $\mu_0 : P \to \mathbb{N}$: *marking* awal

Firing rule: Transisi t enabled jika semua input places memiliki token, dan firing mengupdate marking:

$$\tilde{\mu}(p) = \begin{cases} \mu(p) - 1 & \text{jika } p \in \pi(t) \\ \mu(p) + 1 & \text{jika } p \in \sigma(t) \\ \mu(p) & \text{lainnya} \end{cases}$$

Coloured Petri Net (CPN)

Extension: Token memiliki **warna** (data values)

Basic Petri Net:

- Token homogen
- Hanya menghitung jumlah

Coloured Petri Net:

- Token membawa data
- Warna = job type, machine ID, dll

Definisi 2.2 (CPN)

CPN = $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \Sigma, C, N, E, G, I)$ dengan:

- Σ: himpunan colour sets
- $C : \mathcal{P} \cup \mathcal{T} \to \Phi(\Sigma)$: colour function
- E, G, I: arc expressions, guards, initial marking

TEG: Subkelas Petri net dengan:

- Setiap place: 1 upstream + 1 downstream transition
- Cocok untuk sistem produksi sekuensial
- Dapat dimodelkan dengan *max-plus algebra*

Timing parameters:

- β_j: **firing time** (waktu proses)
- α_i: **holding time** (waktu tunggu minimum)
- w_i: **lag time** (delay awal)

Keunggulan

TEG dapat diubah menjadi *choice-free net* melalui *unfolding*, yang memungkinkan representasi dalam max-plus algebra.

Definisi 3.1 (Max-Plus Algebra)

Max-plus algebra adalah semiring $(\mathbb{R} \cup \{-\infty\}, \oplus, \otimes)$ dengan:

$$a \oplus b = \max(a, b)$$
$$a \otimes b = a + b$$

Elemen identitas:

- $\varepsilon = -\infty$ untuk $\oplus$
- $e = 0$ untuk $\otimes$

Contoh:

$$3 \oplus 5 = \max(3, 5) = 5$$
$$3 \otimes 5 = 3 + 5 = 8$$

Perkalian Matriks Max-Plus

Perkalian matriks dalam max-plus algebra:

$$(A \otimes B)_{ij} = \bigoplus_{k=1}^{n} (a_{ik} \otimes b_{kj}) = \max_{k}(a_{ik} + b_{kj})$$

Contoh:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$(A \otimes B)_{11} = \max(1+5, 2+7) = \max(6, 9) = 9$$

Penting!

Operasi max memodelkan **sinkronisasi** (event hanya terjadi setelah semua prerequisite selesai), dan operasi plus memodelkan **delay**.

TEG dalam Max-Plus Algebra

Dater representation TEG:

$$x^d(k+1) = A \otimes x^d(k) \oplus B \otimes u^d(k)$$

Di mana:

- $x^d(k) \in \mathbb{R}_{\max}^{|Q|}$: state vector (firing times)
- $u^d(k) \in \mathbb{R}_{\max}^{|\mathcal{U}|}$: input vector
- $A \in \mathbb{R}_{\max}^{|Q|\times|Q|}$: state matrix
- $B \in \mathbb{R}_{\max}^{|Q|\times|\mathcal{U}|}$: input matrix

Extended form:

$$\tilde{x}^d(k+1) = \tilde{A}(k)\tilde{x}^d(k) \oplus \tilde{B}(k)\tilde{u}^d(k)$$

dengan memory M untuk menangani delay.

Hubungan Petri Net dan Neural Network

Teorema 4.1

State equation TEG dalam dater representation ekuivalen dengan *two-layer maxout network*.

Bukti (sketch):

1. Maxout network: $h_i(x) = \max_{j \in [0,|Q|]} z_{ij}$ dengan $z_{ij} = W_{ij} v_j + s_{ij}$
2. Set weights $W = 1 \to z_{ij} = v_j + s_{ij}$
3. Dalam max-plus: $h_i(v) = \bigoplus_{j=0}^{|Q|} (s_{ij} \otimes v_j)$
4. Apply ke state dan input: $H = S \otimes v$
5. Substitusi $S \to A$ dan $v \to x$ atau $S \to B$ dan $v \to u$
6. Element-wise max pooling $\to$ final output

Arsitektur Tropical Neural Network

Komponen:

- **State Network**: memproses current state $x(k)$
- **Input Network**: memproses control input $u(k)$
- **Hard-max unit**: element-wise max pooling
- **Output**: next state $x(k+1)$

Activation Function

Hard-max unit (bukan softmax!):

$$h_i(x) = \max_{j \in [0,|Q|]} z_{ij}$$

Kenapa hard-max?

- Hanya satu weight yang berkontribusi per output
- Dapat melacak *path* untuk backpropagation
- Interpretable: menunjukkan bottleneck/critical path

Catatan

Gumbel-Softmax distribution dapat digunakan sebagai alternatif, namun tidak memberikan improvement signifikan (disebutkan di paper).

Problem Statement

Given:

- TEG structure (jumlah places dan transitions)
- Dataset: $\mathcal{D} = \{(x^v(k), u^v(k), x^v(k+1))\}_{v=1}^N$

Goal: Pelajari matriks A dan B dengan meminimalkan loss:

$$\min_{A,B} \mathcal{L}_1 = \min_{A,B} \sum_{v=1}^{N} |\hat{x}(k+1) - x_i(k+1)|$$

Challenge:

- Non-differentiable max operation
- Perlu modifikasi backpropagation

Algorithm 1: Supervised Learning

Algorithm Petri net supervised learning

Require: Dataset $\mathcal{D} = \{(X, u, X')\}$
Ensure: State matrix A, Input matrix B

1: $A \leftarrow -\mathbf{1}^{|Q|\times|Q|}$, $B \leftarrow -\mathbf{1}^{|Q|\times|\mathcal{U}|}$
2: **for** epoch $= 1$ to N **do**
3:     **for** each (X, u, X') in $\mathcal{D}$ **do**
4:         $X' \leftarrow \textsc{ForwardPass}(X, u)$
5:         error $\leftarrow \mathcal{L}_1(X', X')$
6:         $\textsc{BackPropagation}(A, B, \text{error})$
7:     **end for**
8: **end for**
9: **return** A, B

Algorithm 2: Forward Pass

Algorithm Forward propagation in max-plus algebra

Require: Input X, control input u
Ensure: Prediction vector Y

for each perceptron p_i^s in $\mathcal{N}_A$ **do**
    activation $\leftarrow \max_{1 \leq j \leq n}(A_{ij} + X_j)$
    $y_i^a \leftarrow f($activation$)$
    Store activation path
end for
for each perceptron p_j^b in $\mathcal{N}_B$ **do**
    activation $\leftarrow \max_{1 \leq j \leq u}(B_{ij} + u_j)$
    $y_i^b \leftarrow f($activation$)$
    Store activation path
end for
for each index i in output Y **do**
    $Y_i \leftarrow \max(y_i^a, y_i^b)$
    Store activation path
end for
return Y

Algorithm 3: Back-Propagation

Algorithm Back-propagation in max-plus algebra

Require: Weight matrices A, B; error; stored activation paths
Ensure: Updated matrices A, B

for each index i in error **do**
  if activation path i belongs to $\mathcal{N}_A$ **then**
    $A[\text{path}_i] \leftarrow A[\text{path}_i] - \eta \cdot \text{error}_i$
  else
    $B[\text{path}_i] \leftarrow B[\text{path}_i] - \eta \cdot \text{error}_i$
  end if
end for
return A, B

Key insight: Karena hard-max, hanya satu weight per output yang di-update (yang menghasilkan max value).

Robot Manufacturing Cell

Case study: Robot manufacturing cell dengan:

- 2 workpiece types (WP_1, WP_2)
- 3 processing stations (S_1, S_2, S_3)
- 2 input buffers (IB_1, IB_2)
- 1 output buffer (OB)

Routing:

- WP_1: $\mathrm{IB}_1 \to \mathrm{S}_1 \to \mathrm{S}_2 \to \mathrm{S}_3 \to \mathrm{OB}$
- WP_2: $\mathrm{IB}_2 \to \mathrm{S}_2 \to \mathrm{S}_3 \to \mathrm{S}_1 \to \mathrm{OB}$

Tabel: Processing times (detik) untuk setiap workpiece di setiap station

Station	**WP**$_1$	**WP**$_2$
IB$_1$	0	-
IB$_2$	-	0
S$_1$	10	30
S$_2$	20	10
S$_3$	30	20
OB	0	0

Transport time matrix T (detik):

$$T = \begin{bmatrix} 0 & 2 & 4 \\ 2 & 0 & 3 \\ 4 & 3 & 0 \end{bmatrix}$$

Dataset Generation

Procedure:

1. Generate random reference matrices A_{ref} dan B_{ref}
2. Freeze matrices
3. Generate random input sequences $u(k)$
4. Compute trajectories: $x(k+1) = A_{\text{ref}} \otimes x(k) \oplus B_{\text{ref}} \otimes u(k)$
5. Collect dataset: $\mathcal{D} = \{(x(k), u(k), x(k+1))\}$

Dataset size: 1000 training samples, 100 test samples

Catatan

Initial state x_0 harus finite values (bukan $-\infty$) agar gradient dapat flow.

Training Results

Observations:

- Loss converges setelah $\sim$1000 samples
- Learning rate $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ tested
- Best: $\alpha = 10^{-2}$

Matrix Reconstruction

Minkowski distance $(p = 2)$:

- $D(A_{\text{learned}}, A_{\text{ref}}) = 2.05$
- $D(B_{\text{learned}}, B_{\text{ref}}) = 0.82$

Matrix B converges lebih cepat karena struktural properties (lebih sederhana).

Prediction vs Ground Truth

Mean Absolute Error (MAE):

- State x_1: 0.15
- State x_2: 0.22
- State x_3: 0.18
- Overall MAE: 0.18

Robustness to Noise

Test: Tambahkan Gaussian noise $\mathcal{N}(0, \sigma^2)$ ke labels

Tabel: Minkowski distances untuk varying noise levels

Noise Level	Matrix Distance	State Distance
$\sigma = 1\%$	2.05	2.82
$\sigma = 5\%$	9.07	9.59
$\sigma = 10\%$	12.98	29.78

Observation: Model robust terhadap noise kecil $(< 5\%)$, degradasi untuk noise besar.

Dynamic Adaptation

Test: Ubah A matrix di tengah training (50% mark)
Result:

- Dynamic case: model beradaptasi ke target baru
- Static case: stuck pada trajectory lama
- Menunjukkan kemampuan online learning

Kontribusi

Paper contributions:

1. Hubungan formal antara Petri net, max-plus algebra, dan neural network
2. Learnable Petri net representation dalam tropical domain
3. Forward-backward propagation algorithm untuk max-plus algebra
4. Parameter-sharing antara dater dan counter representations
5. Aplikasi pada production scheduling dengan hasil promising

Key Insight

TEG state equation $=$ tropical neural network dengan hard-max units

Keunggulan

Advantages:

- **Interpretable**: Matrix A, B memiliki makna fisik (processing & transport times)
- **Learnable**: Dapat dipelajari dari data observasi
- **Modular**: Hierarki job $\rightarrow$ operation $\rightarrow$ system
- **Scalable**: Extend matriks A, B untuk multiple jobs
- **Adaptive**: Dapat beradaptasi dengan dynamic changes

Limitasi dan Future Work

Limitations:

- Hanya untuk TEG (choice-free nets)
- Initial state harus finite (tidak boleh $-\infty$)
- Convergence speed bergantung pada learning rate

Future work:

- Integrasi dengan reinforcement learning (RL agent)
- Extend ke stochastic Petri nets
- Temperature gradient method untuk speed up
- Real-world deployment pada smart manufacturing