Algoritma Iterasi *Policy* dalam Aljabar Max-Plus

Subiono[a] dan Kistosil Fahim[b]

[a,b]Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
subiono2008@matematika.its.ac.id[a] dan kfahimt@gmail.com[b]

Abstrak-Dalam paper ini dibahas suatu algoritma yang dinamakan Iterasi *Policy*. Algoritma ini tidak hanya dapat digunakan untuk menentukan nilai-eigen dan vektor-eigen dari suatu matriks persegi dalam aljabar max-plus, tetapi juga dapat menentukan *eigenmode* tergeneralisasi. Untuk tujuan komputasi, algoritma yang diberikan diimplementasikan dalam suatu toolbox min-max-plus algebra menggunakan *open source* Scilab v.5.5.2.

Kata kunci: aljabar max-plus; iterasi policy; nilai dan vektor eigen; eigenmode

1 Pendahuluan

Peranan nilai eigen dan vektor eigen dalam aljabar max-plus sangat penting, khususnya untuk perancangan masalah penjadwalan yang teratur. Keteraturan suatu jadwal sangat dibutuhkan oleh pengguna, misalnya pengguna transportasi yang telah disediakan.

Beberapa hasil penjadwalan sistem transportasi sudah dibahas. Khususnya masalah penjadwalan busway, analisis penjadwalan pesawat di suatu bandara dan penjadwalan terintegrasi monorail dan trem ([7, 8, 9] dan [10]). Selain itu masalah penjadwalan rantai pasok dan penjadwalan dari monorail dan train di kota Surabaya menggunakan aljabar max-plus telah dibahas di [11] dan [12]. Keuntungan menggunakan aljabar max-plus adalah umumnya model matematika yang didapat berbentuk linier sedangkan dalam aljabar konvensional taklinier.

Dalam paper ini dibahas beberapa algoritma untuk menentukan nilai eigen dan vektor eigen dari matriks persegi dalam aljabar max-plus. Berbagai power algoritma dibahas terlebih dahulu dan dilanjutkan pembahasan algoritma iterasi *policy*. Hasil pembahasan algoritma diimplementasikan dalam suatu toolbox min-max-plus algebra menggunakan *open source* Scilab v.5.5.1.

2 Aljabar Max-Plus, Graf Berarah dan Berbobot

Dalam bagian ini secara ringkas dikenalkan pengertian aljabar max-plus dan beberapa notasi terkait yang digunakan dalam pebahasan. Pembahasan rinci tentang aljabar max-plus bisa didapat di [1, 2] dan [3].

2.1 Max-plus Algebra

Aljabar max-plus didefinisikan sebagai $\mathbb{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$, dimana $\mathbb{R}_\varepsilon = \mathbb{R} \cup \{\varepsilon\}$ dengan $\mathbb{R}$ adalah himpunan bilangan riil, $\varepsilon \stackrel{\text{def}}{=} -\infty$, $x \oplus y \stackrel{\text{def}}{=} \max\{x, y\}$ dan $x \otimes y \stackrel{\text{def}}{=} x + y$ untuk setiap $x, y \in \mathbb{R}_\varepsilon$. Lagipula, dalam konteks aljabar max-plus pangkat $a^{\otimes^b} = ab$, adalah perkalian a dan b dalam aljabar konvensional. Struktur aljabar dari $(\mathbb{R}_\varepsilon, \oplus, \otimes)$ adalah semi-field idempoten yaitu , semi-ring komutatif idempoten dimana setiap elemen $x \neq \varepsilon$

mempunyai invers $-x$ terhadap operasi $\otimes$ dan untuk setiap $x \in \mathbb{R}_\varepsilon$ berlaku $x \oplus x = x$ ([1]). Untuk ringkasnya semi-ring komutatif idempoten $(\mathbb{R}_\varepsilon, \oplus, \otimes)$ ditulis sebagai $\mathbb{R}_{\max}$. Contoh, dalam $\mathbb{R}_{\max}$; $8 \oplus 5 = \max\{8, 5\} = 8$ dan $4 \otimes 6 = 4 + 6 = 10$. Juga $\varepsilon \oplus x = \max\{-\infty, x\} = x$ dan $0 \otimes x = 0 + x = x$ untuk setiap x di $\mathbb{R}_\varepsilon$ dan $6^{\otimes^7} = 7(6) = 42$.

2.2 Matriks atas $\mathbb{R}_{\max}$

Dalam sub-bagian ini dikenalkan matriks atas $\mathbb{R}_{\max}$. Himpunan semua matriks berukuran $m \times n$ dalam aljbar max-plus dinotasikan oleh $\mathbb{R}_\varepsilon^{m \times n}$. Untuk $n \in \mathbb{N}$ dengan $n \neq 0$, $\underline{n} \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$. Elemen baris ke-$i$ kolom ke-j dari suatu matriks $A \in \mathbb{R}_\varepsilon^{m \times n}$ dinotasikan oleh $a_{i,j}$ untuk $i \in \underline{m}$ dan $j \in \underline{n}$. Adakalanya elemen $a_{i,j}$ juga dinotasikan sebagai $[A]_{i,j},\ i \in \underline{m},\ j \in \underline{n}$. Matriks identitas berukuran $n \times n$ dalam $\mathbb{R}_{\max}$ dinotasikan oleh E, yaitu matriks yang elemen-elemen diagonal utamanya adalah $0 = (e)$ dan elemen yang lainnya sama dengan ε. Jumlahan dari matriks $A, B \in \mathbb{R}_\varepsilon^{m \times n}$ dalam aljabar max-plus dinotasikan sebagai $A \oplus B$ dan didefinisikan oleh

$$[A \oplus B]_{i,j} = a_{i,j} \oplus b_{i,j} = \max\{a_{i,j}, b_{i,j}\}. \tag{1}$$

Untuk $A \in \mathbb{R}_\varepsilon^{m \times n}$ dan skalar $\alpha \in \mathbb{R}_\varepsilon$ perkalian $\alpha \otimes A$ didefinisikan sebagai

$$[\alpha \otimes A]_{i,j} \stackrel{\text{def}}{=} \alpha \otimes a_{i,j} \tag{2}$$

untuk $i \in \underline{m}$ dan $j \in \underline{n}$. Bila $A \in \mathbb{R}_\varepsilon^{m \times p}$ dan $B \in \mathbb{R}_\varepsilon^{p \times n}$, perkalian matriks $A \otimes B$ didefinisikan oleh

$$[A \otimes B]_{i,j} = \bigoplus_{k=1}^{p} a_{i,k} \otimes b_{k,j} = \max_{k \in \underline{p}} \{a_{i,k} + b_{k,j}\}, \tag{3}$$

untuk $i \in \underline{m}$ dan $j \in \underline{n}$. Perkalian matriks dalam aljabar max-plus serupa dengan perkalian matriks dalam aljabar konvensional dimana masing-masing $+$ dan $\times$ diganti oleh $\oplus$ dan $\otimes$.

2.3 Max-Plus dan Graf

Dalam sub-bagian ini dirangkum beberapa definisi dan pengertian terkait dari beberapa hasil-hasil yang didapat di [1, 2].

Definisi 2.1 *Suatu graf berarah $\mathcal{G}$ adalah pasangan $(\mathcal{N}, \mathcal{D})$ dimana $\mathcal{N}$ adalah himpunan titik dari graf $\mathcal{G}$ dan $\mathcal{D} \subseteq \mathcal{N} \times \mathcal{N}$ adalah himpunan simpul dari graf $\mathcal{G}$.*

Definisi 2.2 *Suatu lintasan (path) p dari titik i ke j dalam suatu graf adalah barisan simpul $p = (i_1, i_2, ..., i_{s+1})$ dengan $i_1 = i$ dan $i_{s+1} = j$ sedemikian hingga masing-masing (i_k, i_{k+1}) adalah suatu simpul dari graf tsb. Notasi $||p||_l = s$ menyatakan panjang dari lintasan p adalah s. Himpunan semua lintasan dari titik i ke j yang mempunyai panjang k dinotasikan oleh $P(i, j, k)$.*

Definisi 2.3 *Suatu graf berarah $\mathcal{G}(A)$ yang dikaitkan dengan suatu matriks $A \in \mathbb{R}_\varepsilon^{n \times n}$ adalah graf dimana himpunan titik adalah $\mathcal{N} = \{1, 2, ..., n\}$ dan himpunan simpul $D = \{(j, i) : a_{i,j} \neq \varepsilon\}$. Dalam hal ini dikatakan bahwa $a_{i,j}$ adalah bobot dari simpul $(j, i) \in \mathcal{D}$.*

Definisi 2.4 *Suatu titik j dikatakan dapat dicapai dari suatu titik i bila ada suatu lintasan dari i ke j. Suatu graf dikatakan strongly connected bila sebarang titik dalam graf tsb. dapat dicapai dari sebarang titik yang lainnya. Suatu matriks $A \in \mathbb{R}_\varepsilon^{n \times n}$ adalah tak-tereduksi bila $\mathcal{G}(A)$ adalah strongly connected.*

Definisi 2.5 *Suatu sirkuit dengan panjang s adalah suatu lintasan tertutup, yaitu suatu lintasan sedemikian hingga $i_1 = i_{s+1}$. Suatu sirkuit yang hanya terdiri dari satu titik dinamakan loop. Suatu sirkuit elementer adalah sirkuit dimana semua titik yang termuat dalam sirkuit yaitu $i_1, i_2, \ldots, i_s$ berbeda.*

Definisi 2.6 *Untuk matriks $A \in \mathbb{R}_\varepsilon^{n\times n}$, maka masing-masing matriks A^+ dan A^* adalah $A^+ = \bigoplus_{k=1}^{\infty} A^k$ dan $A^* = E \oplus A^+ = \bigoplus_{k\geq 0} A^{\otimes k}$.*

2.4 Nilai-eigen dan Vektor-eigen

Dalam sub-bagian ini diberikan pengertian dari nilai-eigen dan vektor-eigen dari suatu matriks $A \in \mathbb{R}_\varepsilon^{n\times n}$ dan diberikan beberapa power algoritma untuk menghitung nilai-eigen dan vektor-eigen dari matriks A.

Definisi 2.7 *Diberikan $A \in \mathbb{R}_\varepsilon^{n\times n}$. Bila $\lambda \in \mathbb{R}_\varepsilon$ dan vektor $\boldsymbol{v} \in \mathbb{R}_\varepsilon^n$ setidaknya memuat satu elemen tidak sama dengan ε, dan*

$$A \otimes \boldsymbol{v} = \lambda \otimes \boldsymbol{v}$$

maka λ dinamakan suatu nilai-eigen dari A dan $\boldsymbol{v}$ adalah vektor-eigen.

Masalah untuk mendapatkan suatu nilai-eigen dan vektor eigen dalam aljabar max-plus banyak muncul dari model persamaan

$$\boldsymbol{x}(k+1) = A \otimes \boldsymbol{x}(k),\ k = 0, 1, 2, \ldots, \tag{4}$$

dimana $\boldsymbol{x}(k) \in \mathbb{R}_\varepsilon^n$ dan $A \in \mathbb{R}_\varepsilon^{n\times n}$.

Berikutnya diberikan beberapa Algorima Power yang dapat digunakan untuk menghitung nilai-eigen dan vektor-eigen dari suatu matriks $A \in \mathbb{R}_\varepsilon^{n\times n}$. Algoritma Power memiliki prosedur yang sederhana dan mudah dipahami. Algoritma Power pertama kali dikenalkan oleh Olsder dan diberikan contohnya tetapi pada saat itu belum ada teori pengembangannya ([4]). Pengembangan teori algoritma Power berturut-turut dilakukan oleh Braker ([5]) dan Subiono ([6]).

Algoritma 2.1 ([4])

1. *Ambil sebarang vektor awal $\boldsymbol{x}(0) \neq \boldsymbol{\varepsilon}$.*
2. *Iterasi (4) sampai ada bilangan bulat p, q dimana $p > q \geq 0$ dan bilangan riil c yang memenuhi $\boldsymbol{x}(p) = c \otimes \boldsymbol{x}(q)$.*
3. *Definisikan sebagai nilai-eigen $\lambda = \frac{c}{p-q}$.*
4. *Definisikan sebagai (calon) vektor-eigen*

$$\boldsymbol{v} = \frac{1}{p-q} \sum_{i=1}^{p-q} \boldsymbol{x}(q+i-1).$$

Contoh 2.1 Misalkan dalam sistem (4) matriks A adalah

$$A = \begin{bmatrix} 3 & 5 \\ 3 & 2 \end{bmatrix} \text{ dan vektor keadaan awal } \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Iterasi Persamaan (4) didapat

$$\begin{array}{cccc} \boldsymbol{x}(0), & \boldsymbol{x}(1), & \boldsymbol{x}(2) = 8 \otimes \boldsymbol{x}(0) & \ldots \\ \downarrow & \downarrow & \downarrow & \\ \begin{bmatrix}0\\0\end{bmatrix}, & \begin{bmatrix}5\\3\end{bmatrix}, & \begin{bmatrix}8\\8\end{bmatrix} = 8 \otimes \begin{bmatrix}0\\0\end{bmatrix}, & \ldots \end{array}$$

Dengan demikian digunakan Algoritma Power 2.1 didapat $p = 2, q = 0$ dan $c = 8$. Jadi sebagai nilai-eigen adalah $\lambda = \frac{c}{p-q} = \frac{8}{2} = 4$. Vektor $\boldsymbol{v}$ dari Algoritma 2.1 diberikan oleh

$$\boldsymbol{v} = \frac{1}{2}(\boldsymbol{x}(0) + \boldsymbol{x}(1)) = \frac{1}{2}(\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}5\\3\end{bmatrix}) = \begin{bmatrix}2\frac{1}{2}\\1\frac{1}{2}\end{bmatrix}.$$

Selanjutnya diselidiki apakah $A \otimes \boldsymbol{v} = \lambda \otimes \boldsymbol{v}$.

$$A \otimes \boldsymbol{v} = \begin{bmatrix}3 & 5\\3 & 2\end{bmatrix} \otimes \begin{bmatrix}2\frac{1}{2}\\1\frac{1}{2}\end{bmatrix} = \begin{bmatrix}6\frac{1}{2}\\5\frac{1}{2}\end{bmatrix} = 4 \otimes \begin{bmatrix}2\frac{1}{2}\\1\frac{1}{2}\end{bmatrix} = \lambda \otimes \boldsymbol{v}.$$

□

Algoritma 2.1 tidak selalu memberikan vektor-eigen sebagaimana yang diharapkan. Oleh karena itu diberikan dua algoritma power yang lainnya.

Algoritma 2.2 ([5])

1. *Gunakan Algoritma 2.1 untuk menyelidiki bahwa* $A \otimes \boldsymbol{v} = \lambda \otimes \boldsymbol{v}$.
2. *Bila* $A \otimes \boldsymbol{v} = \lambda \otimes \boldsymbol{v}$, *maka* $\boldsymbol{v}$ *adalah suatu vektor-eigen dari sistem (4) untuk nilai-eigen* λ, *algoritma berhenti. Bila tidak, definisikan vektor baru* $\bar{\boldsymbol{v}}$ *sebagai berikut*

$$\bar{v}_i = \begin{cases} v_i & \text{bila } (A \otimes \boldsymbol{v})_i = \lambda \otimes v_i, \\ \varepsilon & \text{untuk yang lain.} \end{cases}$$

3. *Ulangi lagi (4) dengan* $\boldsymbol{x}(0) = \bar{\boldsymbol{v}}$ *sampai ada beberapa bilangan bulat* $r \geq 0$ *yang memenuhi* $\boldsymbol{x}(r+1) = \lambda \otimes \boldsymbol{x}(r)$. *Maka* $\boldsymbol{x}(r)$ *adalah suatu vektor-eigen dari sistem (4) untuk nilai-eigen* λ.

Algoritma 2.3 ([6])

1. *Ambil sebarang vektor awal* $\boldsymbol{x}(0) \neq \boldsymbol{\varepsilon}$.
2. *Iterasi (4) sampai ada bilangan bulat* p, q *dimana* $p > q \geq 0$ *dan bilangan riil* c *yang memenuhi* $\boldsymbol{x}(p) = c \otimes \boldsymbol{x}(q)$.
3. *Definisikan sebagai nilai-eigen* $\lambda = \frac{c}{p-q}$.
4. *Definisikan sebagai vektor-eigen*

$$\boldsymbol{v} = \bigoplus_{i=1}^{p-q} \left(\lambda^{\otimes(p-q-i)} \otimes \boldsymbol{x}(q+i-1) \right).$$

Contoh 2.2 Misalkan dalam sistem (4) matriks A adalah

$$A = \begin{bmatrix} \varepsilon & 3 & \varepsilon & 1 \\ 2 & \varepsilon & 1 & \varepsilon \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix} \quad \text{dan vektor keadaan awal } \boldsymbol{x}(0) = \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}.$$

Iterasi (4) didapat

$$\begin{array}{cccccc} \boldsymbol{x}(0) & \boldsymbol{x}(1) & \boldsymbol{x}(2) & \boldsymbol{x}(3) & \boldsymbol{x}(4) = 5 \otimes \boldsymbol{x}(2) & \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \begin{bmatrix} 0 \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix} & \begin{bmatrix} \varepsilon \\ 2 \\ 1 \\ \varepsilon \end{bmatrix} & \begin{bmatrix} 5 \\ 2 \\ 4 \\ 2 \end{bmatrix} & \begin{bmatrix} 5 \\ 7 \\ 6 \\ 5 \end{bmatrix} & \begin{bmatrix} 10 \\ 7 \\ 9 \\ 7 \end{bmatrix} & = 5 \otimes \begin{bmatrix} 5 \\ 2 \\ 4 \\ 2 \end{bmatrix}. \end{array}$$

Dengan demikian dalam tiga algoritma power didapat $p = 4, q = 2$ dan $c = 5$. Jadi sebagai nilai-eigen adalah $\lambda = \frac{c}{p-q} = \frac{5}{2} = 2\frac{1}{2}$. Vektor $\boldsymbol{v}$ dari Algoritma 2.1 diberikan oleh

$$\boldsymbol{v} = \frac{1}{2}(\boldsymbol{x}(2) + \boldsymbol{x}(3)) = \frac{1}{2}(\begin{bmatrix} 5 \\ 2 \\ 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 5 \\ 7 \\ 6 \\ 5 \end{bmatrix}) = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix}.$$

Selanjutnya diselidiki apakah $A \otimes \boldsymbol{v} = \lambda \boldsymbol{v}$.

$$A \otimes \boldsymbol{v} = \begin{bmatrix} \varepsilon & 3 & \varepsilon & 1 \\ 2 & \varepsilon & 1 & \varepsilon \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7 \\ 6 \end{bmatrix} \neq \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 7\frac{1}{2} \\ 6 \end{bmatrix} = 2\frac{1}{2} \otimes \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ 5 \\ 3\frac{1}{2} \end{bmatrix} = \lambda \otimes \boldsymbol{v}.$$

Jadi vektor $\boldsymbol{v}$ yang dihasilkan oleh Algoritma 2.1 bukan suatu vektor-eigen dari matriks A untuk nilai-eigen $\lambda = 2\frac{1}{2}$.

Sekarang vektor $\boldsymbol{v}$ dihitung menggunakan Algoritma 2.2, hal ini menghasilkan vektor

$$\bar{\boldsymbol{v}} = \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ \varepsilon \\ 3\frac{1}{2} \end{bmatrix}.$$

Lakukan iterasi ulang dalam (4) dengan vektor awal $\boldsymbol{x}(0) = \bar{\boldsymbol{v}}$ sampai ada beberapa bilangan bulat $r \geq 0$ yang memenuhi $\boldsymbol{x}(r+1) = \lambda \otimes \boldsymbol{x}(r)$, didapat

$$\begin{array}{cccccc} \boldsymbol{x}(0) & \boldsymbol{x}(1) & \boldsymbol{x}(2) & \boldsymbol{x}(3) = 2\frac{1}{2} \otimes \boldsymbol{x}(2) & & \\ \downarrow & \downarrow & \downarrow & \downarrow & & \\ \begin{bmatrix} 5 \\ 4\frac{1}{2} \\ \varepsilon \\ 3\frac{1}{2} \end{bmatrix} & \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ \varepsilon \end{bmatrix} & \begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{bmatrix} & \begin{bmatrix} 12\frac{1}{2} \\ 12 \\ 11\frac{1}{2} \\ 10 \end{bmatrix} & = 2\frac{1}{2} \otimes \begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{bmatrix}. \end{array}$$

Terlihat bahwa $r = 2$ dan $\boldsymbol{x}(2)$ adalah vektor-eigen dari A untuk nilai-eigen $\lambda = 2\frac{1}{2}$. Selanjutnya dari hasil iterasi awal yang telah dilakukan dalam (4) digunakan Algoritma 2.3

didapat vektor $\boldsymbol{v}$ yang diberikan oleh

$$\boldsymbol{v} = (\lambda \otimes \boldsymbol{x}(2)) \oplus \boldsymbol{x}(3) = \begin{bmatrix} 7\frac{1}{2} \\ 4\frac{1}{2} \\ 6\frac{1}{2} \\ 4\frac{1}{2} \end{bmatrix} \oplus \begin{bmatrix} 5 \\ 7 \\ 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ 5 \end{bmatrix}$$

Dapat diselidiki bahwa $\boldsymbol{v}$ adalah vektor-eigen dari A untuk nilai-eigen $\lambda = 2\frac{1}{2}$ sebagai berikut:

$$A \otimes \boldsymbol{v} = \begin{bmatrix} \varepsilon & 3 & \varepsilon & 1 \\ 2 & \varepsilon & 1 & \varepsilon \\ 1 & 2 & 2 & \varepsilon \\ \varepsilon & \varepsilon & 1 & \varepsilon \end{bmatrix} \otimes \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ 5 \end{bmatrix} = \begin{bmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{bmatrix} = 2\frac{1}{2} \otimes \begin{bmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ 5 \end{bmatrix} = \lambda \otimes \boldsymbol{v}.$$

2.5 Algoritma Iterasi *policy*

Algoritma yang efisien untuk menghitung komponen eigen dari sistem linier $(\max, +)$ adalah algoritma iterasi *policy* yang diuraikan oleh Cochet-Terrasson ([13]). Algoritma didapatkan dari algoritma iterasi *multichain policy* oleh Howard ([14]) yang mana telah diketahui secara umum merupakan teori dari Markov. Pada [13] diperkenalkan algoritma generalisasi eigenmode yaitu pasangan $(\boldsymbol{\eta}, \boldsymbol{v})$ dengan $\boldsymbol{\eta}, \boldsymbol{v} \in \mathbb{R}^n$. Jika ada $M \in \mathbb{R}$ sedemikian hingga

$$\text{untuk } m \in \mathbb{R}, m \geq M \Longrightarrow D^{\otimes^m} \otimes \boldsymbol{v} = \mathcal{A}(D^{\otimes^{-1}}) \otimes D^{\otimes^m} \otimes \boldsymbol{v}, \tag{5}$$

dimana D adalah $\text{diag}(\eta_1, \eta_2, ..., \eta_n)^T$, $D^{\otimes^m}$ adalah $\text{diag}(m \times \eta_1, ..., m \times \eta_n)$ dan

$$\mathcal{A}(\lambda) = \bigoplus_{t=0}^{l} A_t \otimes \lambda^{\oplus^t}.$$

Jika $\mathcal{A}$ mempunyai nilai eigen yaitu λ_0 maka Persamaan 5 dapat dituliskan sebagai

$$\boldsymbol{v} = \mathcal{A}(\lambda_0^{\otimes^{-1}}) \otimes \boldsymbol{v}$$

atau

$$\bigoplus_{t=0}^{l} A_t \otimes \lambda_0^{\otimes^{-t}} \otimes \boldsymbol{v} = \boldsymbol{v}.$$

Jika $\mathcal{A}$ tidak punya nilai eigen, maka algoritma iterasi *policy* menghitung vektor $\boldsymbol{\eta} = (\eta_1, \eta_2, ..., \eta_n)^T$ dimana setiap η_i berhubungan dengan maksimum nilai eigen tergenalisir dari kelas yang mengakses simpul q_i. Seperti halnya eigen vektor tergenalisir $\boldsymbol{v}$, Persamaan 5 dapat dituliskan sebagai

$$(D^{\otimes^m})_{j,j} \otimes v_j = \bigoplus_{t=0}^{l} \bigoplus_{i=1}^{n} (((A_t)_{j,i} - (D^{\otimes^t})_{i,i}) \otimes (D^{\otimes^m})_{i,i} \otimes v_i),$$

atau

$$m \times \eta_j + v_j = \max_{t=0,1,...,l} \max_{i=1,2,...,n} (((A_t)_{j,i} - t \times \eta_i) + (m \times \eta_i)) + v_i \tag{6}$$

bagi kedua sisi Persamaan 6 dengan m, selanjutnya untuk $m \to \infty$ didapat

$$\eta_j = \max_{(i,t,j) \in E} \eta_i,$$

dengan $E = \{(i,t,j) \in \mathcal{N} \times \mathcal{N} \times \mathcal{N} | (A_t)_{j,i} \neq \varepsilon\}$ dengan (i,t,j) sisi dari simpul q_i ke q_j dengan t dapat dianggap sebagai banyaknya kendaraan yang melalui lintasan q_i ke q_j atau dalam istilah Petri net dinamakan banyaknya *token* dalam suatu *place*. Jika simpul i mempunyai akses terhadap simpul j maka η_i sama dengan η_j. Maka Persamaan 6 bisa dituliskan sebagai

$$v_j = \max_{(i,t,j)\in\bar{E}} (\tau_{ji} - \mu_{ji} \times \eta_i + v_i)$$

dengan τ_{ji} dan μ_{ji} berturut-turut adalah waktu (bobot) dan banyaknya kendaraan (token) yang melalui lintasan dari i ke j. Sebelum dijelaskan algoritma iterasi *policy*, pertama diberikan asumsi berikut ini.

Asumsi 1 *Diasumsikan bahwa matriks polinomial* $\mathcal{A}(\gamma) = \bigoplus_{t=0}^{l} A_t \otimes \gamma^{\otimes^t} \in \mathbb{R}_{\varepsilon}^{n\times n}$ *mempunyai paling sedikit satu elemen berhingga pada tiap barisnya dan graf dengan bobot waktu* A_0 *yang dinotasikan sebagai* $\mathcal{G}_{TM_0}$ *tidak memiliki sirkuit.*

Telah diketahui bahwa jika $\mathcal{A}$ memenuhi Asumsi 1 didapat eigenmode tergeneralisir $(\boldsymbol{\eta}, \boldsymbol{v})$, maka *cycle time vector* dari $\mathcal{A}$ adalah $\chi(\mathcal{A}) = \boldsymbol{\eta}$. Algoritma iterasi *policy* mengkonstruksi sebuah subgraf dari $\mathcal{G}_{TM}$ sedemikian hingga setiap simpul (j) mempunyai tepat satu simpul (i) dengan (i,t,j) terdapat pada subgraf tersebut. Catatan bahwa subgraf ini memuat paling sedikit satu sirkuit. Himpunan simpul yang terhubung dengan setiap simpul dalam subgraf tersebut berkaitan dengan yang dinamakan *policy*. Untuk definisi yang lebih formal diberikan sebagai berikut.

Algoritma iterasi *policy* biasanya berisi dua tahap untuk setiap iterasi ke-k. Bagian pertama, menghitung pasangan $(\boldsymbol{\eta}^k, \boldsymbol{x}^k)$ yang berhubungan dengan *policy* yang diberikan, tahap ini disebut sebagai tahap **penentuan nilai**. Bagian kedua adalah bagian untuk menentukan *policy* yang lebih baik berdasarkan sikel rata-rata dari sirkuit atau berdasarkan vektor $\boldsymbol{x}^k$ yang berdasarkan pada subgraf, tahap ini disebut sebagai tahap **perbaikan *policy***.

Secara formal, policy adalah pemetaan $\pi : V \to E$ (dengan V himpunan simpul) didefinisikan sebagai

$$\pi(j) = (i,t,j) = p_{j,i}^t \text{ dimana } i \in P_j = \{s | p_{j,s}^t \in E\}$$

untuk semua $j \in V$. Dinotasikan input dari titik pada suatu *policy* dengan $In(\pi j)$ dan bobot waktu dari sisi yang menghubungkan kedua simpul diberikan oleh $\tau(\pi(j)) := \tau_{j,In(\pi(j))}$ dan inisialisasi banyaknya kendaraan diberikan oleh $\mu(\pi(j)) := \mu_{j,In(\pi(j))}$. Selanjutnya *policy* π dikaitkan dengan matriks polinomial $\mathcal{A}^\pi(\gamma) = \bigoplus_{t=0}^{l} A_t^\pi \otimes \gamma^{\oplus^t}$, dimana

$$(A_t^\pi)_{ji} = \tau(\pi(j)), \text{ jika } i = In(\pi(j)) \text{ dan } t = \mu(\pi(j))$$

dan ε untuk lainnya.

Matriks $A^\pi(\gamma)$ mempunyai tepat satu entri berhingga perbaris yang berhubungan terhadap output simpul dari simpul yang dipilih pada π. Timed marked Graph (TMG) $\mathcal{G}_{TM}^\pi$ yang berhubungan dengan $\mathcal{A}^\pi(\gamma)$ adalah subgraf dari $\mathcal{G}_{TM}$ dengan setiap simpul mempunyai simpul tunggal yang terhubung terhadapnya. Jelas bahwa $\mathcal{G}_{TM}^\pi$ memuat paling sedikit satu sirkuit.

Pada tahap penentuan nilai, dihitung vektor $\boldsymbol{\eta}^k$ dan $\boldsymbol{x}^k$ berdasarkan *policy* π sedemikian hingga

$$\begin{aligned} x_j^k &= \bigoplus_{i=1}^{n} (\mathcal{A}^\pi((\eta_i^k)^{\otimes^{-1}}))_{j,i} \otimes x_i^k \\ &= \bigoplus_{i=1}^{n} (A_{\mu(\pi(j))})_{j,i} \otimes (\eta_i^k)^{\otimes^{\mu(\pi(j))}} \otimes x_i^k \\ &= \tau(\pi(j)) - \mu(\pi(j)) \times \eta_i^k + x_{In(\pi(j))} \end{aligned} \tag{7}$$

untuk semua $j = 1, 2, ..., n$. Persamaan 7 berdasarkan definisi dari *policy*, untuk setiap simpul q_j mempunyai hanya satu simpul yang terhubung dalam $\mathcal{G}^{\pi}_{TM}$. Persamaan 7 diselesaikan untuk $(\boldsymbol{\eta}^k, \boldsymbol{x}^k)$. Pertama tentukan sirkuit dalam $\mathcal{G}^{\pi}_{TM}$ berdasarkan *policy* π yang diberikan. Selanjutnya hitung rata-rata sikel $\bar{\eta}$ dari sirkuit yang ditentukan. Pilih sebarang simpul q_i pada sirkuit dan tentukan waktu sikel $\eta_i^k = \bar{\eta}$ dan nilai untuk $x_i^k = x_i^{k-1}$. Setiap simpul q_j yang terhubung terhadap *path* dari q_i dalam $\mathcal{G}^{\pi}_{TM}$ ditentukan waktu sikel $\bar{\eta}$ dan nilai x_j^k dihitung sesuai Persamaan (7). Proses ini dilakukan untuk semua sirkuit dalam *policy* π.

Pada tahap perbaikan *policy*, dilakukan perbaikan *policy* berdasarkan $(\boldsymbol{\eta}^k, \boldsymbol{x}^k)$ yang telah didapat pada tahap sebelumnya. Pertama, untuk setiap verteks q_j periksa apakah ada sisi $p_{j,i}^t$ dengan input simpul q_i mempunyai waktu sikel yang lebih besar dari η_i^k. Jika ada, *policy* $\pi(j)$ dirubah ke sisi $p_{j,i}^t$ yang menghubungkan q_j ke sirkuit dengan sikel rata-rata yang lebih besar. JIka *policy* tidak dapat diperbaiki dengan cara ini maka pilih $\boldsymbol{x}^k$ yang bisa memperbaiki. Jika ada q_j sedemikian hingga

$$\bigoplus_{i=1}^{n} (\mathcal{A}((\eta_i^k)^{\otimes^{-1}}))_{j,i} \otimes x_i^k > x_j^k, \tag{8}$$

maka *policy* π tidak optimal. Perbaikan policy yang ditentukan oleh pengubahan $\pi(j)$ untuk setiap komponen j yang memenuhi (8) dan untuk sisi $p_{j,i}^t$ yang mana bagian kiri dari Persamaan 8 maksimum. Untuk algoritma iterasi *policy* diberikan sebagai berikut.

Algoritma Iterasi Policy
Input:Polinomial matriks $\mathcal{A}$ memenuhi Asumsi 1.
Output: Generalisasi eigenmode dari $\mathcal{A}$

1. **Inisialisasi**
   Pilih sebarang *policy* π_1. Setting $k = 1$ dan $\boldsymbol{x}^0 = (0, ..., 0)^T \in \mathbb{R}^n$. Tentukan pasangan $(\boldsymbol{\eta}^1, \boldsymbol{x}^1)$ menggunakan tahap 3 dari algoritma

2. **Perbaikan Policy**
   Misal

$$\begin{aligned} J &= \{j | \max \eta_i^k > \eta_j^k\}, \\ K(j) &= \arg\max_{p_{j,i} \in P} \eta_i^k, j = 1, 2, ..., n, \\ I &= \{j | \max_{p_{j,i} \in K(j)} (\tau_{ji} - \mu_{j,i}\tau_i^k + x_i^k) > x_j^k\}, \\ L(j) &= \arg\max_{p_{j,i} \in K(j)} (\tau_{j,i} - \mu_{j,i}\tau_i^k + x_i^k), j = 1, 2, ..., n. \end{aligned}$$

   jika $I = J = \emptyset$ maka berhenti. Generalisasi *eigenmode* diberikan oleh $(\boldsymbol{\eta}^k, \boldsymbol{x}^k)$. Lebih lanjut, jika $J \neq \emptyset$. Maka setting

$$\pi_{k+1}(j) = \begin{cases} \text{pilih } p_{j,i} \in K(j) & \text{jika } j \in J \\ \pi_k(j) & \text{jika } j \notin J, \end{cases}$$

   dan jika tidak ($J = \emptyset$ tetapi $I \neq \emptyset$) setting

$$\pi_{k+1}(j) = \begin{cases} \text{pilih } p_{j,i} \in L(j) & \text{jika } j \in I \\ \pi_k(j) & \text{jika } j \notin I. \end{cases}$$

3. **Penentuan Nilai**
   Tentukan sirkuit ξ pada graf $\mathcal{G}_{TM}^{\pi_k}$ dan misal

$$\bar{\eta} = \frac{\sum_{p_{uv}\in\xi} \tau_{uv}}{\sum_{p_{uv}\in\xi} \mu_{uv}}$$

   pilih sebarang simpul $q_i \in \xi$ dan setting $\eta_i^k = \bar{\eta}, x_i^k = x_i^{k-1}$. Untuk simpul j yang terhubung terhadap *path* i pada $\mathcal{G}_{TM}^{\pi_k}$, setting

$$\begin{aligned} \eta_j^k &= \bar{\eta} \\ x_j^k &= \tau(\pi(j)) - \mu(\pi(j)) \times \bar{\eta} + x_{In(\pi(j))}^k. \end{aligned}$$

   Jika ada himpunan takkosong C dari simpul q_i yang takterhubung dengan q_i, maka lakukan kembali tahap 3 menggunakan sirkuit yang berbeda dari sebelumnya.

4. **Iterasi *Policy***
   setting $k = k + 1$ dan menuju ke tahap 2.

Contoh 2.3

Diberikan matriks polinomial

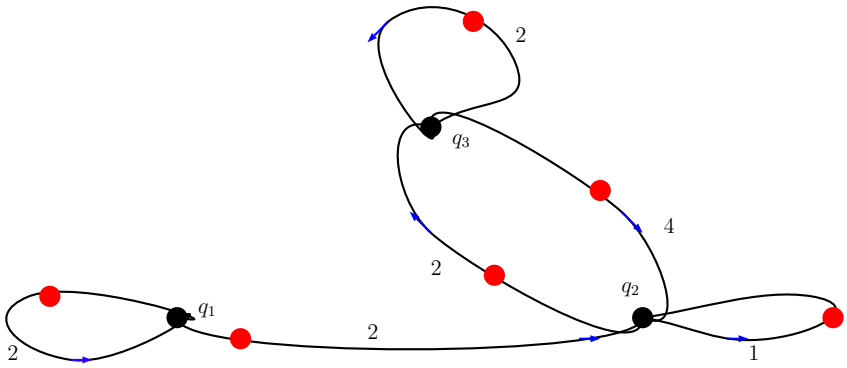
$$\mathcal{A}(\lambda) = A_0 + A_1\lambda$$

dengan

$$A_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, A_1 = \begin{bmatrix} 2 & 2 & \varepsilon \\ \varepsilon & 1 & 4 \\ \varepsilon & 2 & 2 \end{bmatrix}.$$

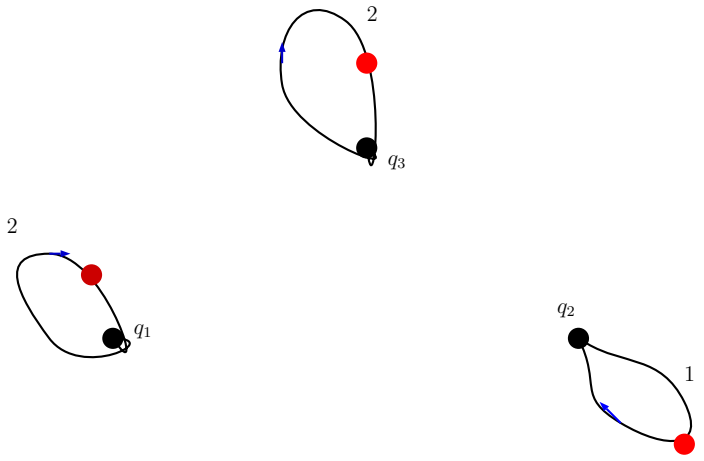
Tentukan nilai eigen dan vektor eigen dari matriks polinomial diatas.

Penyelesaian
Sebelum menggunakan algoritma iterasi *policy* terlebih dahulu digambarkan graf $\mathcal{G}_{TM}$ dari matriks polinomial diatas.



Gambar 1: Graf dari polinomial pada Contoh 2.3

selanjutnya diaplikasikan algoritma iterasi *policy* yaitu
Tahap Inisialisasi
pilih $\boldsymbol{x}^0 = (0, 0, 0)^T$ dan $\pi_1 = \{p_{11}^1, p_{22}^1, p_{33}^1\}$ dapat digambar graf dari $\mathcal{G}_{TM}^{\pi_1}$ yaitu dapat dilihat pada Gambar 2.
Tahap Penentuan Nilai
Ambil sikel $q_1 \to q_1$ pada $\mathcal{G}_{TM}^{\pi_1}$ sehingga $\bar{\eta} = \frac{2}{1} = 2$ didapat $x_1^1 = 0$ dan $\eta_1^1 = \bar{\eta} = 2$. Karena tidak ada simpul lain yang terhubung ke q_1 maka ambil sikel lain di $\mathcal{G}_{TM}^{\pi_1}$ yaitu $q_2 \to q_2$

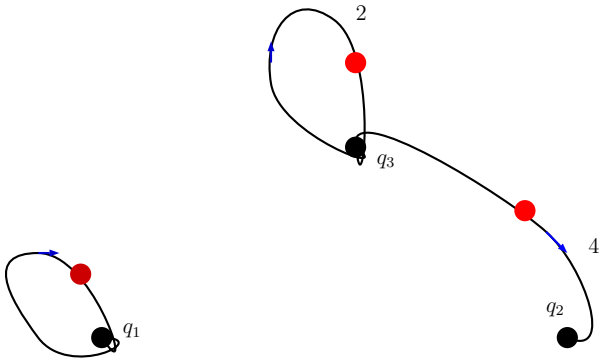


Gambar 2: Graf dari *policy* pada iterasi ke-1

sehingga didapat $x_2^1 = 0$ dan $\eta_2^1 = 1$ selanjutnya dengan cara yang sama didapat $x_3^1 = 0$ dan $\eta_3^1 = 2$, Dengan demikian pada tahap ini didapatkan $\boldsymbol{x}^1 = (0,0,0)^T, \boldsymbol{\eta}^1 = (2,1,2)^T$.

Perbaikan *policy*

Dari Gambar 1 dan nilai $\boldsymbol{x}^1$ dan $\boldsymbol{\eta}^1$ didapatkan $J = \{2\}$, $K(1) = \{1\}$, $K(2) = \{3\}$, $K(3) = \{3\}$, $I = \{2\}$, $L(1) = \{1\}$, $L(2) = \{3\}$, $L(3) = \{3\}$. Karena $J \neq \emptyset$ maka didapat $\pi_2 = \{p_{11}^1, p_{2,3}^1, p_{33}^1\}$ (untuk gambar graf dari $\mathcal{G}_{TM}^{\pi_2}$ dapat dilihat pada Gambar 3).



Gambar 3: Graf dari *policy* pada iterasi ke-2

Tahap Penentuan Nilai

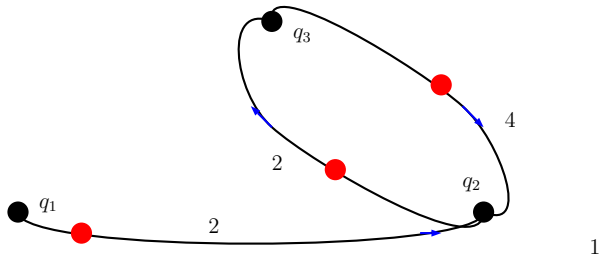
Ambil sikel $q_1 \to q_1$ pada $\mathcal{G}_{TM}^{\pi_2}$ sehingga $\bar{\eta} = \frac{2}{1} = 2$ sehingga didapat $x_1^2 = 0$ dan $\eta_1^2 = \bar{\eta} = 2$. Karena tidak ada simpul lain yang terhubung ke q_1 maka ambil sikel lain di $\mathcal{G}_{TM}^{\pi_1}$ yaitu $q_3 \to q_3$ sehingga didapat $x_3^2 = 0$ dan $\eta_3^2 = 2$. Karena q_2 terhubung terhadap q_3 maka didapat $\eta_2^2 = 2$ dan $x_2^3 = \tau(p_{23}^1) - \mu(p_{23}^1)\bar{\eta} + x_3^2 = 4 - 1 \cdot 2 + 0 = 2$. Jadi $\boldsymbol{x}^2 = (0,2,0)^T, \boldsymbol{\eta}^1 = (2,2,2)^T$.

Perbaikan *policy*

Dari Gambar 1 dan nilai $\boldsymbol{x}^1$ dan $\boldsymbol{\eta}^1$ didapatkan $J = \emptyset$, $K(1) = \{1,2\}$, $K(2) = \{2,3\}$, $K(3) = \{2,3\}$, $I = \{1,3\}$, $L(1) = \{2\}$, $L(2) = \{3\}$, $L(3) = \{2\}$. Karena $J = \emptyset$ dan $I \neq \emptyset$ maka didapat $\pi_3 = \{p_{12}^1, p_{23}^1, p_{32}^1\}$ (untuk gambar graf dari $\mathcal{G}_{TM}^{\pi_3}$ dapat dilihat pada Gambar 4).

Tahap Penentuan Nilai

Ambil sikel $q_2 \to q_3 \to q_2$ pada $\mathcal{G}_{TM}^{\pi_3}$ sehingga $\bar{\eta} = \frac{4+2}{1+1} = 2$ kemudian ambil simpul q_3 sehingga didapat $x_3^3 = 0$ dan $\eta_3^3 = \bar{\eta} = 3$. Karena q_1 dan q_2 terhubung terhadap q_3 maka didapat $\eta_1^3 = 3$, $\eta_2^3 = 3$, $x_2^3 = \tau(p_{23}^1) - \mu(p_{23}^1)\bar{\eta} + x_3^3 = 4 - 1 \cdot 3 + 0 = 1$ dan $x_1^3 = \tau(p_{12}^1) - \mu(p_{12}^1)\bar{\eta} + x_2^3 = 2 - 1 \cdot 3 + 1 = 0$. Jadi $\boldsymbol{x}^2 = (0,1,0)^T, \boldsymbol{\eta}^1 = (3,3,3)^T$.



Gambar 4: Graf dari *policy* pada iterasi ke-3

Perbaikan *policy*

Dari Gambar 1 dan nilai dari $\boldsymbol{x}^1$ dan $\boldsymbol{\eta}^1$ didapatkan $J = \emptyset$, $K(1) = \{1, 2\}$, $K(2) = \{2, 3\}$, $K(3) = \{2, 3\}$, $I = \emptyset$, $L(1) = \{1, 2\}$, $L(2) = \{3\}$, $L(3) = \{2, 3\}$. Karena $J = \emptyset$ dan $I = \emptyset$ maka iterasi berhenti. Sehingga didapat eigenmode tergeneralisir adalah

$$\boldsymbol{\eta} = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}, \ \boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Algoritma ini telah diimplementasikan pada toolbox Scilab berikut ini contohnya.

```
--> A=[2,2,-%inf;-%inf,1,4;-%inf,2,2];

-->[eigvalMode,x] = policyIteration(A)

Number Of Iteration

3.
x  =

2.
3.
2.
eigvalMode  =

3.
3.
3.

-->\\Atau kita bisa merubahnya kedalam bentuk polynomial sebagai berikut

-->A0=maxpluszeros(3,3);

-->A1=[2,2,-%inf;-%inf,1,4;-%inf,2,2];

-->A=A0;

-->A(:,:,2)=A1
A  =

(:,:,1)
```

```
- Inf  - Inf  - Inf
- Inf  - Inf  - Inf
- Inf  - Inf  - Inf
(:,:,2)

2.     2.  - Inf
- Inf    1.    4.
- Inf    2.    2.

-->[eigvalMode,x] = policyIteration(A)

Number Of Iteration

3.
x  =

2.
3.
2.
eigvalMode  =

3.
3.
3.
```

Dari Contoh-contoh yang telah dibahas menunjukkan bahwa semua elemen vektor *cycle mean* adalah sama dengan λ. Berbeda dengan contoh-contoh sebelumnya, contoh berikut menghasilkan elemen vektor *cycle mean* ada yang berbeda.

Contoh 2.4 Diberikan matriks

$$A = \begin{bmatrix} 8 & \varepsilon & \varepsilon \\ 13,5 & 5 & 5 \\ 33,5 & 25 & 25 \end{bmatrix}.$$

Gunakan Algoritma Iterasi *policy* didapat *eigenmode* tergenaralisir $(\boldsymbol{\eta}, \boldsymbol{v})$ dimana

$$\boldsymbol{\eta} = \begin{bmatrix} 8 \\ 25 \\ 25 \end{bmatrix} \quad \text{dan} \quad \boldsymbol{v} = \begin{bmatrix} 0 \\ 5,5 \\ 25,5 \end{bmatrix}.$$

Dapat ditunjukkan bahwa $\boldsymbol{\eta}$ dan $\boldsymbol{v}$ memenuhi

$$A \otimes (k \times \boldsymbol{\eta} + \boldsymbol{v}) = (k+1) \times \boldsymbol{\eta} + \boldsymbol{v}, \quad k \geq 0.$$

3 Kesimpulan dan Penelian Berikutnya

Telah dibahas beberapa Algoritma Power. Ada tiga Algoritma Power. Ketiga algoritma ini memberikan kondisi : untuk sebarang keadaan awal $\boldsymbol{x}(0) \neq \boldsymbol{\varepsilon}$, iterasi (4) sampai ada bilangan bulat p, q dimana $p > q \geq 0$ dan bilangan riil c yang memenuhi $\boldsymbol{x}(p) = c \otimes \boldsymbol{x}(q)$. Kondisi ini ekivalen dengan semua elemen dari vektor *cycle mean* sama dengan $\lambda = \frac{c}{p-q}$. Bila kondisi ini tidak dipenuhi, maka Algoritma 2.1, 2.2 dan Algoritma 2.3

gagal memberikan hasil yang diharapkan. Walaupun Algoritma 2.1 memenuhi kondisi yang telah dibicarakan, adakalanya algoritma ini tidak menghasilkan vektor $\boldsymbol{v}$ sebagai vektor-eigen dari matriks A untuk nilai-eigen $\lambda = \frac{c}{p-q}$. Maka untuk menentukan vektor $\boldsymbol{v}$ sebagai vektor-eigen dari A untuk nilia-eigen $\lambda = \frac{c}{p-q}$ digunakan Algoritma 2.2.

Berbeda dengan dua Algoritma 2.1 dan Algoritma 2.2. Algoritma 2.3 dapat langsung digunakan tanpa menggunakan Algoritma 2.1 dan menghasilkan vektor $\boldsymbol{v}$ sebagai vektor-eigen dari matriks A untuk nilai-eigen $\lambda = \frac{c}{p-q}$.

Lain halnya dengan tiga Algoritma Power, Algoritma Iterasi *Policy* tidak perlu memenuhi syarat yang harus dipenuhi oleh Algoritma 2.1, Algoritma 2.2 dan Algoritma 2.3. Tetapi Algoritma Iterasi *Policy* memberikan hasil-hasil yang lebih general dari apa yang dihasilkan oleh tiga Algoritma Power. Apapun hal ini, dari pembahasan menunjukkan bahwa prosedur-prosedur yang dilakukan dalam Algoritma Iterasi *Policy* jauh lebih rumit dan kompleks dibandingkan tiga Algoritma Power. Oleh karena itu untuk penelitian berikutnya bila memungkinkan, maka Algoritma 2.3 akan dikembangkan dan diharapkan memberikan hasil-hasil yang sama dalam Algoritma Iterasi *Policy*.

Daftar Pustaka

[1] Subiono, *Aljabar Min-Max Plus dan Terapannya*, Jurusan Matematika, Institut Teknologi Sepuluh Nopember, 2015.

[2] B. Heidergot, G.J.Olsder, J.Woude, Max Plus at Work, *Princeton Universty Press, New Jersey*, 2006.

[3] Baccelli F., Cohen G., Olsder G.J., Quadrat J.P., Synchronization and Linearity. An Algebra for Discrete Event Systems, *Wiley, London, 489 pp*, 1992. Versi Web dapat didownload di https://www.rocq.inria.fr/metalau/cohen/documents/BCOQ-book.pdf

[4] G.J. Olsder, "Eigenvalues of dynamical min-max systems", *Journal of Discrete Event Dynamical Systems*, 1:177-207, (1991).

[5] J.G. Braker, "Algorithms and Applications Timed Discrete Event Systems", Ph.D thesis, Department of Technical Mathematics and Informatics Delft University of Technology, (1993).

[6] Subiono, "On classes of min-max-plus systems and their applications", Ph.D thesis, Department of Technical Mathematics and Informatics Delft University of Technology, (2000).

[7] Winarni and Subiono, On Scheduling City Bus Routes using Max Plus Algebra. *Proceedings of The National Seminar on Mathematics IV, Sepuluh Nopember Institute of Technology*, 2008.

[8] Nahlia Rakhmawati, Subiono and Subchan, Busway Schedule Planning In Surabaya Using Max-Plus Algebra. *Proceedings of National Graduate Seminar XII - ITS, Surabaya*, July 12, 2012.

[9] Dyah Arum Anggraeni, Subchan and Subiono, Analysis of Aircraft Transit Timetable in Airport Using Max-Plus Algebra, *Journal POMITS Vol. 1, No. 1*, (2013), page: 1-5

[10] Fatma Ayu Nuning F.A., Modeling and Scheduling Integrated System of Monorail and Train in Surabaya City Using Max-Plus Algebra, *Final Project, Mathematics Department Faculty of Mathematics and Science, Sepuluh Nopember Institute of Technology*, 2013.

[11] Ema Enggar Wati, Subiono, and Subchan, Analysis Scheduling of Supply Chain Using Max-Plus Algebra (Case Study Terminal Bahan Bakar Minyak (TBBM) Manggis, Bali), *Journal POMITS Vol. 1, No.1*, (2014), page: 1-6

[12] Kistosil Fahim, Lukman Hanafi, Subiono, Fatma Ayu, Monorail and Tram Scheduling Which Integrated Surabaya Using Max-Plus Algebra. *Proceedings of ISIM-MED 2014, Yogyakarta*, 2014.

[13] Cochet-Terrasson, J., Cohen, G., Gaubert, S., Mc Gettrick, M., Quadrat, J.P., "Numerical Computation of Spectral Elements in (max,+) Algebra", IFAC Conference on System Structure and Control, Nantes, France, 1998.

[14] Howard, R.A., "Dynamic Programming and Markov Processes", MIT Press/Wiley, New York, 1960.