Max-plus algebra

Een toepassing op tramlijn 19

D. Visser

TU Delft

Max-plus algebra

Een toepassing op tramlijn 19

door

D. Visser

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging
van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathtematics

Studentnummer:	4380223	
Projectduur:	23 februari 2018 – 21 juni, 2018	
Beoordelingscommissie:	Dr. J.W. van der Woude,	TU Delft, supervisor
	Prof. dr. ir. H.X. Ling,	TU Delft
	Dr. J.L.A. Dubbeldam,	TU Delft
	Drs. E.M. van Elderen,	TU Delft

Op dit verslag is geheimhouding van toepassing tot en met 21 juni 2018.

TU Delft

Samenvatting

In dit onderzoek hebben we een nieuwe dienstregeling voor tramlijn 19 opgesteld. Deze tramlijn rijdt nu tussen Delft en Leidschenhage. In 2020 zal het traject worden uitgebreid met haltes op de campus van de TU Delft. In dit onderzoek hebben we twee dienstregelingen opgesteld: één voor in de spitsuren, en één voor in de daluren. Er zijn een aantal voorwaarden meegenomen in het onderzoek. Zo moet de dienstregeling aansluiten op de collegetijden van de studenten die gebruik maken van tram 19. Ook moeten we de voorwaarden die we in de huidige dienstregeling vinden, meenemen in het nieuwe traject. Denk bijvoorbeeld aan de frequentie waarin trams rijden, of wanneer de dienstregeling start op een dag.
Daarnaast hebben we onderzocht hoe het beste kan worden overgegaan van een spitsuur dienstregeling naar een met daluren.
Dit alles is gedaan met behulp van max-plus algebra, waarbij het nemen van een maximum en optellen, de conventionele operaties van optellen en vermenigvuldigen vervangen. Het power algoritme helpt ons bij het kiezen van de juiste vertrektijden, zodat we een constant vertrekschema krijgen. Ook maken we gebruik van Petrinetten om het gehele netwerk te modelleren.
In dit onderzoek wordt steeds eerst gekeken naar een kleiner of versimpeld tramnetwerk. Hierna passen we de theorie toe op het toekomstige traject. Op het einde hebben we een dienstregeling kunnen opstellen voor tramlijn 19. In deze dienstregeling vindt afwisseling plaats tussen spits-en daluren, met elk een andere dienstregeling en hoeveelheid trams. Hierbij is rekening gehouden met de voorwaarden die wij voor ons model hebben opgesteld.

Voorwoord

Om heel eerlijk te zijn zag ik tijdens mijn studie Technische Wiskunde nogal op tegen het Bachelor Eind Project (BEP). Ik kon mij toen nog niet voorstellen dat ik een heel kwartaal lang zelfstandig een wiskundig probleem zou gaan aanpakken. Daarnaast was ik bang dat ik onderzoek doen heel saai zou vinden. De studie is af en toe best abstract en ik word niet heel opgewonden van het bewijzen van ingewikkelde stellingen of het oplossen van vele sommen die de verbeelding voorbij schieten. Ik was daarom bang dat het BEP ook heel abstract en "droog"zou worden. Uiteindelijk bleek dat ik ook zelf een onderwerp mocht aandragen en ik heb toen gekozen voor dit modelleer project over tramlijn 19.
Dit zou ik natuurlijk niet allemaal in mijn voorwoord zetten als dit waar bleek te zijn. Ik moet bekennen dat ik het erg leuk vond om dit project te doen. Het was heel plezierig om voor een langere tijd je tanden ergens in te zetten. Het was ook fijn dat dit project erg praktisch is: tramlijn 19 bestaat echt en moet inderdaad uitgebreid worden. Ik vond het heel prettig om theoretische wiskunde te gebruiken voor praktische toepassingen.
In de eerste gesprekken met mijn begeleider gaf ik aan dat ik graag meer vaardigheid in Python wilde krijgen d.m.v. dit project. Dat is gelukt. Alle berekeningen zijn gedaan in Python en ik heb in een korte tijd het programma beter leren kennen. Na het tekenen van meerdere Petrinetten kan ik zeggen dat ik ook redelijk behendig ben geworden in Adobe Illustrator©.
Als laatste zou ik graag mijn begeleiders Jacob van der Woude en Hai Xiang Ling willen bedanken. Zij gaven mij gedurende deze weken veel ondersteuning en energie. Na besprekingen met hen kon ik altijd weer fris aan de slag om verder te werken.
Ik hoop dat u als lezer dit verslag met net zo veel plezier leest als dat ik eraan heb gewerkt!

D. Visser
Delft, juni 2018

Inhoudsopgave

1

Introductie max-plus algebra

Het doel van dit Bachelor Eind Project is het opstellen van een dienstregeling voor tramlijn 19. Hierbij maken we gebruik van een soort algebra, de zogehete *max-plus algebra*. Kort gezegd is dit een gebied binnen de wiskunde waar we in plaats van dingen plus (+) en keer (×) elkaar doen, we respectievelijk het maximum nemen en optellen. In dit hoofdstuk wordt dit nog verder toegelicht.
Max-plus algebra behoort niet tot de stof die wij in de bachelor Technische Wiskunde leren. Het lijkt enigzins op de gewone, lineaire algebra, maar kent een paar kenmerkende verschillen.
Max-plus algebra blijkt een uiterst handige methode om vertrektijden van voertuigen in een bepaald netwerk te bepalen. Hoewel in dit onderzoek dit niet echt van toepassing is, is max-plus algebra bijvoorbeeld heel geschikt om dienstregelingen te bepalen van verschillende (soorten) voertuigen die onderling een bepaalde "relatie"met elkaar hebben. Denk bijvoorbeeld aan een sprinter die moet wachten op de passagiers van een intercity voordat hij vertrekt vanaf het station.
Een aantal nadelen kent de max-plus algebra ook: zo is het niet mogelijk om de voertuigen in het systeem te laten versnellen (dit zullen we later tegenkomen). Ook kan de andere notatie in het begin wat verwarrend zijn. Hieronder vindt u daarom ook een uitgebreid hoofdstuk met de basis definites en eigenschappen die bij max-plus algebra horen.
Naast deze nadelen kent de max-plus algebra echter ook één groot voordeel: het werk heel intuïtief. Soms zo intuïef dat het bijna niet meer op wiskunde, maar meer op logisch verwoorden lijkt! Dit maakt het echter zo sterk in haar toepassing op dit soort dienstregelingen: de berekeningen kunnen direct gelinkt worden aan de praktijk. Dit zal duidelijker worden in hoofdstukken 2 en 3.
Hieronder volgen een aantal basiseigenschappen en definites die bij max-plus algebra horen. Schrik niet: er volgt hierna een voorbeeld waarin de rekenregels duidelijk worden.

1.1. Basiseigenschappen en notaties max-plus algebra

1.1.1. Algemene definities en notaties

In de max-plus algebra zijn de additieve (+ en −) en multiplicatie operatoren(× en ÷) vervangen door respectievelijk het nemen van het maximum ($\oplus$) en optellen ($\otimes$). Dit geeft ons de volgende definites:

- Additiviteit:

  $x \oplus y = max\{x, y\}$

- Multiplicatie:

  $x \otimes y = x + y$

Verder definiëren we $\epsilon = -\infty$ en $e = 0$. We noteren de verzameling $\mathbb{R}_{max} = \mathbb{R} \cup \{\epsilon\}$. Nemen we de elementen x, y, z, ϵ, e uit de verzameling $\mathbb{R}_{max}$, c uit $\mathbb{R}$ en n uit $\mathbb{N}$, dan krijgen we de volgende algebraïsche eigenschappen:

- Associativiteit:

$$
\begin{aligned}
x \oplus (y \oplus z) &= max\{x, max\{y, z\}\} \\
&= max\{z, max\{x, y\}\} \\
&= (x \oplus y) \oplus z \\
x \otimes (y \otimes z) &= x + (y + z) \\
&= (x + y) + z \\
&= (x \otimes y) \otimes z
\end{aligned}
$$

- Commutativiteit:

$$\begin{aligned} x \oplus y &= max\{x, y\} \\ &= max\{y, x\} \\ &= y \oplus x \\ x \otimes y &= x + y \\ &= y + x \\ &= y \otimes x \end{aligned}$$

- Neutraal element:

$$\begin{aligned} x \oplus \epsilon &= max\{x, -\infty\} \\ &= x \\ x \otimes e &= x + 0 \\ &= x \end{aligned}$$

- Distributiviteit van $\otimes$ over $\oplus$:

$$\begin{aligned} x \otimes (y \oplus z) &= x + max\{y, z\} \\ &= max\{x + y, \quad x + z\} \\ &= (x \otimes y) \oplus (x \otimes z) \end{aligned}$$

- Machtsverheffen:

$$\begin{aligned} x^{\otimes n} &= \underbrace{x \otimes x \otimes \cdots \otimes x}_{n \text{ keer}} \\ &= \underbrace{x + x + \cdots + x}_{n \text{ keer}} \\ &= n \times x \end{aligned}$$

1.1.2. Uitbreiding naar vectoren en matrices

Voor vectoren en matrices gelden dezelfde regels, maar volgt een iets andere notatie. Voor een $n \times m$ matrix wordt de onderliggende max-plus algebra genoteerd als $\mathbb{R}_{max}^{n \times m}$. Voor $i, j \in \mathbb{N}$, met $i = \{1, \ldots n\}$ en $j = \{1, \ldots, m\}$ noteren we a_{ij} voor een element uit de i^e rij en de j^e kolom. We kunnen zo'n matrix A dan als volgt schrijven:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}$$

De additieve operatie $\oplus$ is voor de matrices $A, B \in \mathbb{R}_{max}^{n,m}$ en $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ gedefinieerd door:

$$\begin{aligned} [A \oplus B]_{ij} &= a_{ij} \oplus b_{ij} \\ &= max\{a_{ij}, b_{ij}\} \end{aligned}$$

Het is hierbij dus belangrijk dat de matrices die de additieve operatie ondergaan dezelfde grootte hebben.
De multiplicative operatie $\otimes$ is voor de $n \times m$ matrix A en $m \times l$ matrix B met $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ en $k \in \{1, \ldots, l\}$ gedefinieerd door:

$$\begin{aligned} [A \otimes B]_{ik} &= \bigoplus_{j=1}^{m} a_{ij} \otimes b_{jk} \\ &= max\{a_{ij} + b_{jk}, j = 1, \ldots, m\} \end{aligned}$$

Hier moeten de matrices die de multiplicative operatie ondergaan niet perse dezelfde grootte hebben, maar wel hetzelfde aantal rijen.

Vervolgens kunnen we nog een vector $\mathbf{x} \in \mathbb{R}_{max}^{n,1}$ met een scalar c uit $\mathbb{R}$ vermenigvuldigen:

$$\begin{aligned} \mathbf{x} \otimes c &= \mathbf{x} + c \\ &= \begin{pmatrix} x_1 + c \\ x_2 + c \\ \vdots \\ x_n + c \end{pmatrix} \end{aligned}$$

Eigenwaarden en eigenvectoren

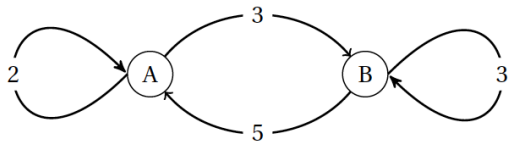
We introduceren verder nog de begrippen eigenwaarden en eigenvectoren. We bekijken een $n \times n$ matrix A met:

$$A \otimes \mathbf{v} = \lambda \otimes \mathbf{v} \tag{1.1}$$

Hierbij is λ een scalar en $\mathbf{v}$ een vector van $n \times 1$ zodanig dat tenminste één element niet-oneindig is. Het i^e element van $\lambda \otimes \mathbf{v}$ is hier gelijk aan $\lambda \otimes v_i = \lambda + v_i$.
Als we een λ en vinden zoals in (1.1), dan noemen we λ de eigenwaarde van matrix A en $\mathbf{v}$ de bijbehorende eigenvector. Deze eigenvector is niet uniek: als we dezelfde constante bij elk element van de eigenvector optellen is dit nog steeds een eigenvector. Dit is vergelijkbaar in de conventionele algebra, waarbij we de eigenvector met een scalar kunnen vermenigvuldigen en het dan nog steeds een eigenvector is.

1.1.3. Voorbeeld trein

Er zal in dit hoofdstuk een voorbeeld worden gegeven met een klein traject van twee stations. Dit is een veelgebruikt voorbeeld en komt uit het boek Max Plus at Work [1]. Dit traject kun je zien in figuur 1.1. Op het traject rijden vier treinen. Op de lokale trajecten van A en B rijdt elk een trein. Daarnaast rijden er nog twee treinen tussen de stations A en B. Het lokale traject van A naar A bedraagt twee minuten reistijd. Het lokale traject van B naar B bedraagt drie minuten reistijd. De reistijd van A naar B bedraagt drie minuten, terwijl de reistijd van B naar A vijf minuten bedraagt.



Figuur 1.1: Een traject met twee treinstations A en B

Er zijn een aantal voorwaarden waaraan dit voorbeeld moet voldoen:

- De reistijd van de treinen op de trajecten is vast
- De frequentie waarin er treinen van de stations A en B vertrekken moet zo hoog mogelijk zijn
- De tijden waarop de treinen van de stations vertrekken moeten regelmatig zijn
- Treinen die aankomen bij een station moeten daar wachten totdat de andere trein van het andere traject is aangekomen op het station om een overstap voor de passagiers mogelijk te maken
- Wanneer een trein kan vertrekken doet deze dat

We noteren de vertrektijden van station A en B respectievelijk met x_A en x_B. De vector $\mathbf{x} \in \mathbb{R}^2$, met $\mathbf{x} = \binom{x_A}{x_B}$ geeft de vertrektijden van beide stations weer. In de vroege ochtend vertrekken de treinen van de stations A en B op het tijdstip $\mathbf{x}(0)$. De eerstvolgende keer dat de treinen vertrekken is op $\mathbf{x}(1)$ enzovoorts. Met de bovenstaande eisen kunnen we nu de vergelijkingen opstellen voor de $k+1^e$ vertrektijd van station A en B:

$$x_A(k+1) \geq x_A(k)+2, \qquad x_A(k+1) \geq x_B(k)+5 \tag{1.2}$$
$$x_B(k+1) \geq x_B(k)+3 \qquad x_B(k+1) \geq x_A(k)+3 \tag{1.3}$$

Omdat er rekening moet worden gehouden met het overstappen door passagiers op de stations van de ene op de andere trein moeten we het maximum nemen van de twee vertrektijden. Op die manier weten we zeker dat er pas een volgende trein vertrekt zodra de reizigers, afkomstig uit de lokale trajecten van A en B, genoeg tijd hebben gehad om over te stappen op een trein van het traject tussen A en B.

$$x_A(k+1) \geq max\{x_A(k)+2, \quad x_B(k)+5\} \tag{1.4}$$
$$x_B(k+1) \geq max\{x_B(k)+3, \quad x_A(k)+3\} \tag{1.5}$$

Aangezien de frequentie van de vertrekken zo hoog mogelijk moet zijn en een trein zodra hij kan vertrekken dit ook doet, kunnen de vergelijkingen als gelijkheden worden geschreven:

$$x_A(k+1) = max\{x_A+2, \quad x_B+5\} \tag{1.6}$$
$$x_B(k+1) = max\{x_B+3, \quad x_A+3\} \tag{1.7}$$

Voor deze vergelijkingen kan ook de matrix $A = \left(\begin{smallmatrix} 2 & 5 \\ 3 & 3 \end{smallmatrix}\right)$ worden opgesteld. Via de volgende vergelijking kunnen we dan de $k+1^e$ vertrektijden berekenen:

$$\mathbf{x}(k+1) = A \otimes \mathbf{x}(k) \tag{1.8}$$

Als eenmaal de beginvector $\mathbf{x}(0)$ is bepaald, staan de vertrektijden hierna ook vast. Stel we nemen in dit geval de beginvector $\mathbf{x}(0) = \mathbf{0}$, dan kunnen we met 1.8 de vertrektijden van A en B voor $k = 1$ berekenen:

$$\begin{aligned}
\mathbf{x}(1) &= A \otimes \mathbf{x}(0) \\
&= \begin{pmatrix} 2 & 5 \\ 3 & 3 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} 2 \otimes 0 \oplus 5 \otimes 0 \\ 3 \otimes 0 \oplus 3 \otimes 0 \end{pmatrix} \\
&= \begin{pmatrix} max\{2+0, & 5+0\} \\ max\{3+0, & 3+0\} \end{pmatrix} \\
&= \begin{pmatrix} 5 \\ 3 \end{pmatrix}
\end{aligned}$$

Wordt dit gedaan voor $k = 0, 1, \ldots$, dan krijgt men de volgende vertrektijden:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \begin{pmatrix} 8 \\ 8 \end{pmatrix}, \begin{pmatrix} 13 \\ 11 \end{pmatrix}, \begin{pmatrix} 16 \\ 16 \end{pmatrix} \tag{1.9}$$

Zoals u ziet zijn de vertrektijden vanaf de stations niet regelmatig. De ene keer vertrekt een trein drie minuten later, de andere keer weer vijf minuten.

Omdat dit voor reizigers lastig te onthouden is, kiezen we een andere begin vector $\mathbf{x}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Met deze beginvector worden de vertrektijden voor $k = 0, 1, \ldots$:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 5 \\ 4 \end{pmatrix}, \begin{pmatrix} 9 \\ 8 \end{pmatrix}, \begin{pmatrix} 13 \\ 12 \end{pmatrix}, \begin{pmatrix} 17 \\ 16 \end{pmatrix}, \tag{1.10}$$

Er vertrekt nu vanaf station A en B elke vier minuten een trein. Dit is voor de reiziger gemakkelijk te onthouden.

Voor een klein traject is het gemakkelijk om gewoonweg uit te proberen welke begin vector "werkt". Voor grotere trajecten is het handiger gebruik te maken van het power algoritme (zie 1.2).

1.2. Power algoritme

Het power algoritme is een methode om een eigenvector en eigenwaarde te vinden. Het wordt in dit onderzoek gebruikt om een goede beginvector van vertrektijden te kiezen, zodat we direct een regelmatig vertrekschema verkrijgen.

Stel we hebben een $n \times n$ matrix A waarop we vergelijking 1.8 herhaaldelijk toepassen. Dan volgt dat we na een paar keer periodiek gedrag zullen zien. D.w.z. dat we na een aantal keer toepassen een veelvoud van een al eerdere vector krijgen. Via het power algoritme wordt er gezocht na hoeveel stappen deze periodiciteit zich voordoet. Hierna kan er een eigenwaarde en eigenvector worden gevonden. Nemen we deze eigenvector als beginvector van een dienstregeling, dan volgen er direct regelmatige vertrektijden. Dit algoritme is dus erg handig om gemakkelijk achter een goede beginvector te komen, wanneer men te maken heeft met wat grotere matrices.

Hieronder volgen de stappen van het power algoritme met erna een voorbeeld uit het boek Max Plus at Work [1].

1.2.1. Power algoritme

1. Neem een willekeurige beginvector $\mathbf{x}(0)$ zodanig dat $\mathbf{x}(0)$ tenminste één niet-eindig element bevat.

2. Herhaal (1.8) totdat er gehele getallen p, q, met $p > q \geq 0$ zijn en een reëel getal c zodanig dat $\mathbf{x}(p) = \mathbf{x}(q) \otimes c$. Er is dan periodiek gedrag bereikt.

3. Bereken de eigenwaarde λ via $\lambda = c/(p-q)$ (dit is de regulier deling, dus geen max-plus algebra deling)

4. Bereken de eigenvector via $\mathbf{v} = \bigoplus_{j=1}^{p-q} \left(\lambda^{\otimes(p-q-j)} \otimes \mathbf{x}(q+j-1)\right)$

1.2.2. Voorbeeld

We passen het power algoritme toe op de matrix $A = \begin{pmatrix} \epsilon & 3 & \epsilon & 1 \\ 2 & \epsilon & 1 & \epsilon \\ 1 & 2 & 2 & \epsilon \\ \epsilon & \epsilon & 1 & \epsilon \end{pmatrix}$.
Als beginvector nemen we $x(0) = (0, \epsilon, \epsilon, \epsilon)^\top$ en passen (1.8) herhaaldelijk toe. De volgende vertrektijden worden dan verkregen:

$$\mathbf{x}(1) = \begin{pmatrix} \epsilon \\ 2 \\ 1 \\ \epsilon \end{pmatrix}, \qquad \mathbf{x}(2) = \begin{pmatrix} 5 \\ 2 \\ 4 \\ 2 \end{pmatrix}, \qquad \mathbf{x}(3) = \begin{pmatrix} 5 \\ 7 \\ 6 \\ 5 \end{pmatrix}, \qquad \mathbf{x}(4) = \begin{pmatrix} 10 \\ 7 \\ 9 \\ 7 \end{pmatrix}$$

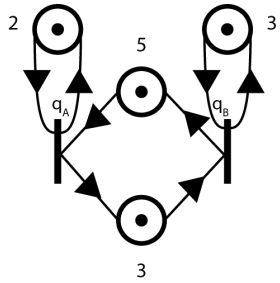
enz.
Zie dat $\mathbf{x}(4) = \mathbf{x}(2) \otimes 5$. Via het power algoritme vinden we nu dus dat $p = 4, q = 2$ en $c = 5$, dus $\lambda = 5/4 - 2 = 2\frac{1}{2}$.
Vervolgens berekenen we de eigenvector $\mathbf{v}$ via het algoritme:

$$\mathbf{v} = (\lambda \otimes \mathbf{x}(2)) \oplus \mathbf{x}(3) = \begin{pmatrix} 7\frac{1}{2} \\ 4\frac{1}{2} \\ 6\frac{1}{2} \\ 4\frac{1}{2} \end{pmatrix} \oplus \begin{pmatrix} 5 \\ 7 \\ 6 \\ 5 \end{pmatrix} = \begin{pmatrix} 7\frac{1}{2} \oplus 5 \\ 4\frac{1}{2} \oplus 7 \\ 6\frac{1}{2} \oplus 6 \\ 4\frac{1}{2} \oplus 5 \end{pmatrix} = \begin{pmatrix} max\{7\frac{1}{2}, 5\} \\ max\{4\frac{1}{2}, 7\} \\ max\{6\frac{1}{2}, 6\} \\ max\{4\frac{1}{2}, 5\} \end{pmatrix} = \begin{pmatrix} 7\frac{1}{2} \\ 7 \\ 6\frac{1}{2} \\ 5 \end{pmatrix}$$

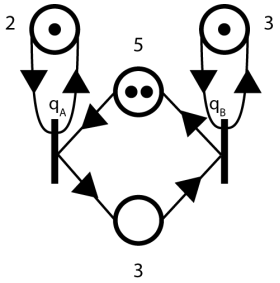
Je kunt deze eigenvector nog gemakkelijk verifiëren door na te gaan dat $A \otimes \mathbf{v} = \begin{pmatrix} 10 \\ 9\frac{1}{2} \\ 9 \\ 7\frac{1}{2} \end{pmatrix} = \lambda \otimes \mathbf{v}$

1.3. Petrinetten

Petrinetten zijn een speciaal soort gerichte grafen. Ze zijn voor het eerst beschreven in het proefschrift van C.A. Petri in 1962. Het is een wiskundige modelleertaal die veel in de informatica wordt gebruikt om activiteiten in een bepaald proces te modelleren. Ook in de max-plus algebra blijken Petrinetten een handige *tool*. Een max-plus model kan worden omgezet in een Petrinetwerk en vice versa.
Een Petrinetwerk heeft een (eindige) verzameling knopen $\mathcal{N}$ die kunnen worden opgededeeld in twee disjuncte verzamelingen $\mathcal{P}$ en $\mathcal{Q}$. De verzameling $\mathcal{P}$ noemen we *plaatsen* en de verzameling $\mathcal{Q}$ noemen we *transities*. Deze plaatsen en transities zijn onderling met elkaar verbonden door pijlen.Er bestaan pijlen van transities naar plaatsen $(q_i \to p_j)$ en van plaatsen naar transities $(p_j \to q_i)$. Pijlen van plaatsen naar plaatsen of van transities naar transities zijn niet mogelijk.
In de plaatsen kunnen zich zogeheten *tokens* bevinden. Het aantal tokens binnen een circuit is constant. In dit onderzoek geven wij de plaatsen aan met rondjes, de transitites met een balkje en een token met een kleiner, ingekleurd rondje. Zie ook figuur 1.2a.



(a) Figuur 1.1 weergeven als Petrinet



(b) Figuur 1.1 na één keer vuren

Figuur 1.2: Petrinetwerk voor en na vuren

De transities en plaatsen kennen we een bepaalde voorwaarde toe voor het doorstromen van de tokens. Zo kunnen plaatsen een bepaalde waarde hebben, denk bijvoorbeeld aan een wachttijd of de trajectduur. Een transitie kan alleen een token doorlaten, als er zich in de plek voor de transitie een token bevindt. Dit doorsturen noemen we ook wel *vuren*.
Om bovenstaande duidelijker te maken, hebben we figuur 1.1 omgeschreven naar een Petrinet. De transities bestaan hier dus uit de twee stations A en B en de plaatsen stellen hier de trajecten tussen A en B en hun lokale trajecten voor. De cijfers hierbij staan voor de tijd dat een trein zich op het traject bevindt. De tokens in de plaatsen staan voor de treinen die zich op het betreffende traject bevinden

In bovenstaand figuur zien we dat voor elke transitie zich een token bevindt in de plaats ervoor. Als we transitie q_B laten vuren, krijgen we het Petrinetwerk zoals in figuur 1.2b

Op het traject van B naar A bevinden zich nu dus twee treinen. Transitie q_B zal in de volgende stap niet zomaar kunnen vuren. Er bevindt zich wel een token in de plaats op het lokale traject van B, maar op de plaats van het traject van A naar B is geen token beschikbaar voor het vuren van transitie q_B. Er zal dus op een vuren van transitie q_A moeten worden gewacht, voordat transitie q_B in een volgende stap weer zou kunnen vuren.

2

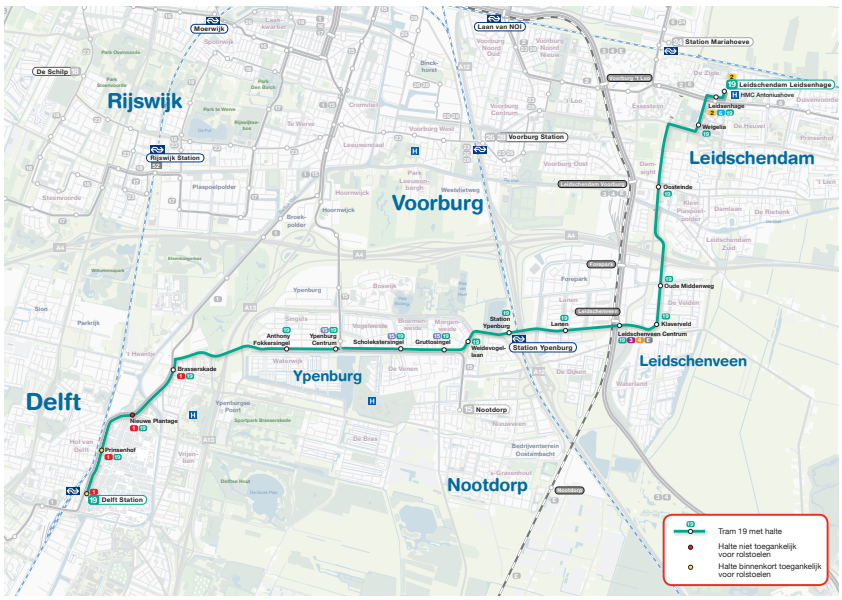
Opstellen dienstregeling

2.1. Inleiding probleem

In dit hoofdstuk zullen we een reisschema opstellen voor tram 19 in de regio Haaglanden. Deze tram bestaat sinds 2010 en rijdt langs verschillende haltes tussen Leidschendam en Delft station. In 2020 moet de dienstregeling uitgebreid worden van Delft station naar de TU Delft campus. Hier komen nu nog verschillende streekbussen langs de halte TU Aula. Tevens is er een bus die specifiek alleen de haltes tussen Delft station en de campus aandoet. Langs deze haltes moet ook tram 19 gaan rijden. Op dit moment wordt nog gewacht op vernieuwing van de Bastiaansbrug in Delft. Deze is nu nog te zwak om tramstellen te dragen.
Daarom wordt in dit onderzoek een dienstregeling voor 2020 opgesteld. Omdat we te maken hebben met meer haltes en een langer traject, zal er het een en ander aan de huidige dienstregeling veranderd moeten worden. Ook moet er met de uitbreiding naar de TU campus goed worden nagedacht over de vertrektijden: voordat de colleges beginnen moeten er genoeg trams langs de TU haltes rijden om studenten vanaf het station naar de universiteit te brengen.

2.1.1. Gegevens

Op dit moment rijdt tram 19 vanaf de beginhalte HMC Anthoniushoeve (Leidschendam) naar het eindpunt Delft station (Delft). Deze reis duurt ± 35 minuten. Het traject kunt u in figuur 2.1 zien. Een grotere afbeelding hiervan is terug te vinden in appendix A.



Figuur 2.1: Huidig traject tramlijn 19

Een overzicht van de haltes in het huidige traject vindt u terug in figuur 2.2. Hierin vindt u ook de haltes die zullen horen bij het uitgebreidde traject in 2020.

(a) Huidige haltes tramlijn 19

(b) Uitbreiding haltes naar TU campus

Figuur 2.2: Huidige en toekomstige haltes tramlijn 19

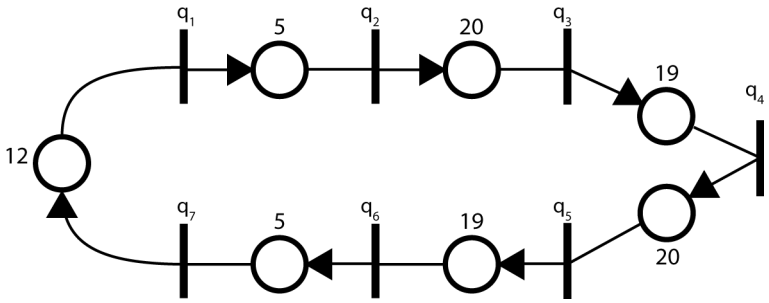
In ons model hebben we de halte station Delft opgedeeld in de haltes 'Delft station aankomst' en 'Delft station vertrek'. Tram 19 wacht hier namelijk enkele minuten, zodat reizigers kunnen overstappen van de trein op de tram. Ook de halte HMC is opgesplits in twee haltes. Dit is de begin-/eindhalte. Om de tram de tijd te geven om alle reizigers te laten uit te stappen en om te keren is hier ook een wachttijd van enkele minuten. Hetzelfde geldt voor de toekomstige eindhalte TU Technopolis.
In de huidige dienstregeling vertrekt de eerste tram om 5:23 vanaf HMC Antoniushoeve richting Delft [6]. Overdag rijdt tramlijn 19 elke 15 minuten, vanaf 20 u 's avonds wordt dat elke 20 minuten [6,7].
Nu we deze gegevens over het huidige traject hebben, kunnen we beginnen met het opstellen van een traject voor tram 19. Dit zal in twee stappen worden gedaan. In hoofdstuk 2.2 zal m.b.v. max-plus algebra een dienstregeling worden opgesteld voor een versimpeld traject. De reden dat we dit doen is omdat het uiteindelijke traject uit zeer veel haltes bestaat en het noodzakelijk is goed te snappen hoe het model bij een kleiner traject werkt, voordat we een dienstregeling voor tramlijn 19 uitgebreid naar de TU kunnen opstellen.

2.2. Versimpeld traject

De haltes van het huidige en toekomstige traject kunt u zien in figuur 2.2.

In het huidige traject komt dit neer op zo'n 18 haltes. Breiden we dit uit naar het toekomstige traject met de TU Campus, dan krijgen we zelfs 25 haltes.
Omdat het traject een circuit is, wordt een halte steeds twee keer aangedaan: op de heenweg en op de terugweg. De haltes hebben niet noodzakelijk dezelfde vertrektijden op de heen-en terugweg. Het is daarom belangrijk dat we ze als afzonderlijke haltes beschouwen. Hiernaast hebben we nog extra haltes toegevoegd: HMC Anthoniushoeve, TU Technopolis en Delft station (zowel op de heen- als terugweg), bestaan uit een aankomst-en vertrekhalte. Dit heeft te maken met de wachttijd die op deze haltes geldt. Dit komt verder aan bod in hoofdstuk 2.3. Al met al betekent dat we voor het toekomstige traject de vertrektijden van zo'n 52 haltes bekijken.
Het versimpelde Petrinetwerk van het toekomtige traject van tramlijn 19 is te zien in figuur 2.3. Hierbij heb-

ben we een groot aantal haltes samengevoegd tot dummi haltes en hun reistijden opgeteld.



Figuur 2.3: Petrinetwerk van het versimpelde traject van tramlijn 19

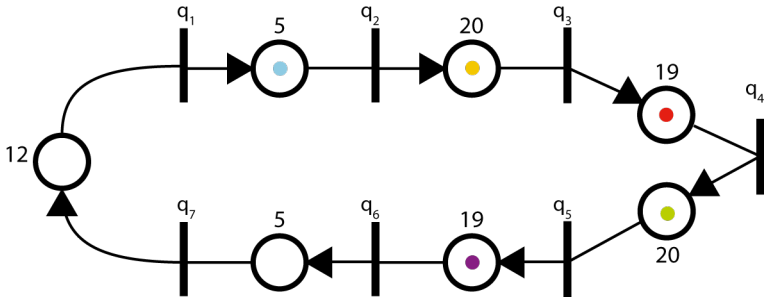
We zien hier dat $\mathscr{Q}$ bestaat uit de verzameling $\{q_1,...,q_7\}$. Deze stellen de haltes voor op ons traject. $\mathscr{P}$ is de verzameling $\{p_1,....,p_7\}$, waarbij p_i staat voor de reistijd van halte q_i naar q_{i+1} en $q_8 = q_1$. Deze trajecttijden staan vast. Er kan dus niet korter worden gereden over een trajectstuk door het versnellen van een tram. Samengevat geeft dit ons de volgende waarden voor p_i en q_j. De waarden van p_i zijn hierbij in minuten.

- q_1 := halte TU Aula
- q_2 := halte Delft Station
- q_3 := dummie halte (overkoepelend voor de haltes tussen Delft Station en HMC Anthoniushoeve)
- q_4 := halte HMC Anthoniushoeve
- q_5 := dummie halte (overkoepelend voor de haltes tussen HMC Anthoniushoeve en Delft Station)
- q_6 := halte Delft Station
- q_7 := halte TU Aula
- $p_1 := 5$
- $p_2 := 20$
- $p_3 := 19$
- $p_4 := 20$
- $p_5 := 19$
- $p_6 := 5$
- $p_7 := 12$

Tellen we de waarden van $p_1,\ldots,p_7$ bij elkaar op dan vinden we een totale trajecttijd van 100 minuten. Dat betekent dat een tram na precies 100 minuten weer terug is bij de halte waar hij begon te rijden.

2.2.1. Opstellen max-plus model versimpeld traject

Nu we ons Petrinetwerk hebben opgesteld, kunnen we een max-plus model opstellen. Hiervoor moeten we wel nog eerst trams in ons circuit plaatsen. Aangezien er in de huidige dienstregeling vijf trams rijden, hebben we ervoor gekozen ook vijf trams in ons traject te plaatsen. Waar we deze trams neerzetten is willekeurig: na het opstellen van ons max-plus model zullen we daarna het power algoritme gebruiken om de beste beginposities van de trams te vinden. We kiezen ervoor tokens te plaatsen in p_1, p_2, p_3, p_4 en p_5. Deze tokens staan voor de trams in ons traject. We geven hierbij elke token een kleur zodat de verschillende trams gemakkelijk te onderscheiden zijn, zie ook figuur 2.4



Figuur 2.4: Petrinetwerk van het versimpelde traject van tramlijn 19 met de beginpositie van vijf trams

Aan de hand van figuur 2.4 kunnen we nu een max-plus model opstellen:

$$x_1(k+1) = x_7(k+1) + 12 \tag{2.1}$$
$$x_2(k+1) = x_1(k) + 5 \tag{2.2}$$
$$x_3(k+1) = x_2(k) + 20 \tag{2.3}$$
$$x_4(k+1) = x_3(k) + 19 \tag{2.4}$$
$$x_5(k+1) = x_4(k) + 20 \tag{2.5}$$
$$x_6(k+1) = x_5(k) + 19 \tag{2.6}$$
$$x_7(k+1) = x_6(k+1) + 5 \tag{2.7}$$

$x_i(k)$ geeft hierbij de waarde voor de k^e vuring vanaf transitie q_i aan, ofwel: x_i is het tijdstip wanneer er voor de k^e keer een tram vanaf halte q_i vertrekt. Vergelijking 2.2 vertelt ons hier dat de $k+1^e$ vertrektijd van q_2 afhangt van de k^e vertrektijd van halte q_1 plus de reistijd tussen de twee haltes van 5 minuten.
Bij vergelijking 2.7 ligt het iets anders: er bevindt zich op dit moment nog geen token in de plaats voor q_7. Hij kan dus nog niet "vuren". q_7 zal dus eerst moeten wachten op de token die vanaf q_6 na de $k+1^e$ stap vertrekt. Die moet op zijn beurt weer wachten op de token die vanaf q_5 komt na het k^e vertrek.
Wel kunnen we vergelijking 2.1 en 2.7 omschrijven naar een vorm die ook afhangt van een k^e vertrektijd van een station, i.p.v. een $k+1^e$ vertrektijd. Dit maakt het opstellen van de matrix van het model later gemakkelijker.
In (2.7) zien we $x_7(k+1) = x_6(k+1) + 5$. Substitueren van $x_6(k+1) = x_5(k) + 19$ geeft ons dan:

$$x_7(k+1) = x_5(k) + 24$$

op dezelfde manier vinden we:

$$x_1(k+1) = x_7(k+1) + 12 = x_5(k) + 36$$

Nu kunnen we ons max-plus model uitdrukken in de standaardvorm van $x(k+1) = A \otimes x(k)$ (zie (1.8) uit hoofdstuk 1). Deze ziet er als volgt uit:

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ x_7(k+1) \end{pmatrix} = \begin{pmatrix} \epsilon & \epsilon & \epsilon & \epsilon & 36 & \epsilon & \epsilon \\ 5 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & 20 & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & 19 & \epsilon & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 20 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 19 & \epsilon & \epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & \epsilon & 24 & \epsilon & \epsilon \end{pmatrix} \otimes \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \\ x_7(k) \end{pmatrix} \tag{2.8}$$

Merk op dat de matrix A veel elementen ϵ bevat. Dit is vergelijkbaar met 0-elementen in de conventionele algebra. We willen dan eigenlijk dat het element a_{ij} van onze matrix A "niets"doet met het j^e element van de vector $\mathbf{x}(k)$. De eerste rij van $A \otimes \mathbf{x}(k)$ geeft bijvoorbeeld:

$$\begin{aligned} x_1(k+1) &= (a_{11} \otimes x_1(k)) \oplus (a_{12} \otimes x_2(k)) \oplus (a_{13} \otimes x_3(k)) \oplus (a_{14} \otimes x_4(k)) \oplus (a_{15} \otimes x_5(k)) \oplus (a_{16} \otimes x_6(k)) \oplus (a_{17} \otimes x_7(k)) \\ &= (\epsilon \otimes x_1(k)) \oplus (\epsilon \otimes x_2(k)) \oplus (\epsilon \otimes x_3(k)) \oplus (\epsilon \otimes x_4(k)) \oplus (36 \otimes x_5(k)) \oplus (\epsilon \otimes x_6(k)) \oplus (\epsilon \otimes x_7(k)) \\ &= max\{(\epsilon + x_1(k)), \quad (\epsilon + x_2(k)), \quad (\epsilon + x_3(k)), \quad (\epsilon + x_4(k)), \quad (36 + x_5(k)), \quad (\epsilon + x_6(k)), \quad (\epsilon + x_7(k))\} \\ &= max\{\epsilon, \quad \epsilon, \quad \epsilon, \quad \epsilon, \quad 36 + x_5(k), \quad \epsilon, \quad \epsilon\} \\ &= max\{-\infty, \quad 36 + x_5(k)\} \\ &= 36 + x_5(k) \end{aligned}$$

We zien dus dat het element ϵ hier als neutraal element werkt. In het vervolg worden dan ook niet alle elementen gelijk aan ϵ weergeven in matrices. Ze wordt dan wel aangeduid met doorlopende punten (zie bijvoorbeeld de matrix A' uit vergelijking 2.17).

Stel we kiezen nu voor $\mathbf{x}(0) = \mathbf{0}$. Dan krijgen we met (2.8) de volgende vertrektijden voor $k = 1, 2, ...$, met de waardes weer in minuten:

$$\mathbf{x}(1)=\begin{pmatrix}36\\5\\20\\19\\20\\19\\24\end{pmatrix},\ \mathbf{x}(2)=\begin{pmatrix}56\\41\\25\\39\\39\\39\\44\end{pmatrix},\ \mathbf{x}(3)=\begin{pmatrix}75\\61\\61\\44\\59\\58\\63\end{pmatrix},\ \mathbf{x}(4)=\begin{pmatrix}95\\80\\81\\80\\64\\63\\83\end{pmatrix},\ \mathbf{x}(5)=\begin{pmatrix}100\\100\\100\\100\\100\\99\\88\end{pmatrix},\ \mathbf{x}(6)=\begin{pmatrix}136\\105\\120\\119\\120\\119\\124\end{pmatrix},\ \mathbf{x}(7)=\begin{pmatrix}156\\141\\125\\139\\139\\139\\144\end{pmatrix},\ \ldots \tag{2.9}$$

Het verschil tussen de vertrektijden van de stations is niet constant. De derde vertrektijd vanaf q_1 ligt op 75 minuten. 20 minuten later, op de vierde vertrektijd zal er weer een tram vertrekken vanaf q_1. Hierna vertrekt echter de eerstvolgende tram vijf minuten later: geen constante dienstregeling dus.

Er is verder nog iets dat opvalt: na vijf stappen treedt er periodiek gedrag op. De vector $\mathbf{x}(6)$ is gelijk aan de vector $\mathbf{x}(1)$, maar bij alle elementen is 100 opgeteld. Dit is niet zo gek: het totale traject is 100 minuten lang, dus na 100 minuten herhalen de vertrektijden zich weer.
Maar er valt nog meer te zien in deze eerste vijf stappen. We merkten al eerder op dat de dienstregeling niet constant was. Elke eerstvolgende vertrektijd had weer een ander tijdsverschil met de vorige vertrektijd. Wanneer we echter kijken naar het gemiddelde verschil van het i^e element van de vectoren $\mathbf{x}(1),\ldots,\mathbf{x}(6)$, vinden we voor elk element een gemiddeld verschil van 20. Kijk bijvoorbeeld naar het derde element $x_3(k+1)$, voor $k=0,1,2,3,4,5$ (we rekenen hier met de conventionele -, + en deling):

$$\begin{aligned}&\frac{(x_3(2)-x_3(1))+(x_3(3)-x_3(2))+(x_3(4)-x_3(3))+(x_3(5)-x_3(4))+(x_3(6)-x_3(5))}{5}\\&=\frac{(39-19)+(44-39)+(80-44)+(100-80)+(119-100)}{5}\\&=\frac{20+5+36+20+19}{5}\\&=20\end{aligned} \tag{2.10}$$

Ook dit valt logisch te beredeneren: het totale traject van begin tot eindpunt is 100 minuten. We rijden met vijf trams, dus gemiddeld zal inderdaad elke 20 minuten een halte worden aangedaan door een tram.
In de volgende stap zullen we straks met behulp van het power algoritme (zie hoofdstuk 1.2) laten zien dat er inderdaad geldt dat de eigenwaarde van de matrix van ons stelsel (2.8) gelijk is aan $\lambda=20$. Daarnaast zullen we hiermee een geschikte beginvector vinden, die ervoor zorgt dat de dienstregeling direct regelmatig verloopt.

2.2.2. Power algoritme toegepast op versimpeld traject

Om met het power algoritme te beginnen kiezen we weer de beginvector $\mathbf{x}(0)=\mathbf{0}$. De vectoren $\mathbf{x}(1),\ldots,\mathbf{x}(7)$ hebben we al berekend in (2.9). We volgen hieronder de stappen van het power algoritme (zie hoofdstuk 1.2.1):

1. Neem een willekeurige beginvector $\mathbf{x}(0)$ zodanig dat $\mathbf{x}(0)$ tenminste één niet-eindig element bevat.

Als beginvector hebben we $\mathbf{x}(0)=\mathbf{0}$ gekozen.

2. Herhaal stap (1.8) totdat er gehele getallen p,q, met $p>q\geq 0$ zijn en een reëel getal c zodanig dat $\mathbf{x}(p)=\mathbf{x}(q)\otimes c$. Er is dan periodiek gedrag bereikt.

In (2.9) zien we dat $\mathbf{x}(6)=\mathbf{x}(1)\otimes 100$. Op $k=6$ heeft ons model dus periodiek gedrag bereikt. We vinden dan $p=6, q=1, c=100$.

3. Bereken de eigenwaarde λ via $\lambda=c/(p-q)$

Dit geeft ons, zoals verwacht, $\lambda=100/(6-1)=20$ (zie 2.10).

4. Bereken de eigenvector via $\mathbf{v}=\bigoplus_{j=1}^{p-q}\left(\lambda^{\otimes(p-q-j)}\otimes\mathbf{x}(q+j-1)\right)$

We berekenen de eigenvector **v**:

$$\mathbf{v} = \bigoplus_{j=1}^{p-q} \left(\lambda^{\otimes(p-q-j)} \otimes \mathbf{x}(q+j-1) \right) \tag{2.11}$$

$$= \bigoplus_{j=1}^{5} \left(\lambda^{\otimes(5-j)} \otimes \mathbf{x}(j) \right) \tag{2.12}$$

$$= 20^{\otimes 4} \otimes \mathbf{x}(1) \oplus 20^{\otimes 3} \otimes \mathbf{x}(2) \oplus 20^{\otimes 2} \otimes \mathbf{x}(3) \oplus 20 \otimes \mathbf{x}(4) \oplus \mathbf{x}(5) \tag{2.13}$$

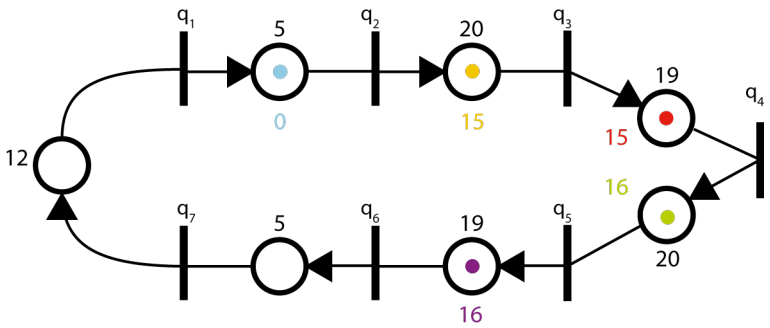
$$= 80 \otimes \begin{pmatrix} 36 \\ 5 \\ 20 \\ 19 \\ 20 \\ 19 \\ 24 \end{pmatrix} \oplus 60 \otimes \begin{pmatrix} 56 \\ 41 \\ 25 \\ 39 \\ 39 \\ 39 \\ 44 \end{pmatrix} \oplus 40 \otimes \begin{pmatrix} 75 \\ 61 \\ 61 \\ 44 \\ 59 \\ 58 \\ 63 \end{pmatrix} \oplus 20 \otimes \begin{pmatrix} 95 \\ 80 \\ 81 \\ 80 \\ 64 \\ 63 \\ 83 \end{pmatrix} \oplus \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 99 \\ 88 \end{pmatrix} \tag{2.14}$$

$$= \begin{pmatrix} 116 \\ 85 \\ 100 \\ 99 \\ 100 \\ 99 \\ 104 \end{pmatrix} \oplus \begin{pmatrix} 116 \\ 101 \\ 85 \\ 99 \\ 99 \\ 99 \\ 104 \end{pmatrix} \oplus \begin{pmatrix} 115 \\ 101 \\ 101 \\ 84 \\ 99 \\ 98 \\ 103 \end{pmatrix} \oplus \begin{pmatrix} 115 \\ 100 \\ 101 \\ 100 \\ 84 \\ 83 \\ 103 \end{pmatrix} \oplus \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 99 \\ 88 \end{pmatrix} \tag{2.15}$$

$$= \begin{pmatrix} 116 \\ 101 \\ 101 \\ 100 \\ 100 \\ 99 \\ 104 \end{pmatrix} := \mathbf{v} \tag{2.16}$$

Eigenvectors zijn niet uniek, en dus kunnen we **v** omschrijven tot de vorm:
$\mathbf{v} = \begin{pmatrix} 17 & 2 & 2 & 1 & 1 & 0 & 5 \end{pmatrix}^{\mathsf{T}}$.

Hierbij hebben we het kleinste element (99) van de vector **v** afgetrokken. We kiezen deze eigenvector **v** als de beginvector **x**(0) van ons stelsel. **x**(0) geeft dan de begin vertrektijden $x_1, ..., x_7$ weer. Dit is dus nog voordat vergelijking (1.8) is toegepast. Hierbij vertrekt vanaf halte q_1 als laatste een tram op het tijdstip $t = 17$. Als we op dit tijdstip een momentopname van ons Petrinetwerk zouden maken met de positie van de trams, krijgen we het volgende te zien:



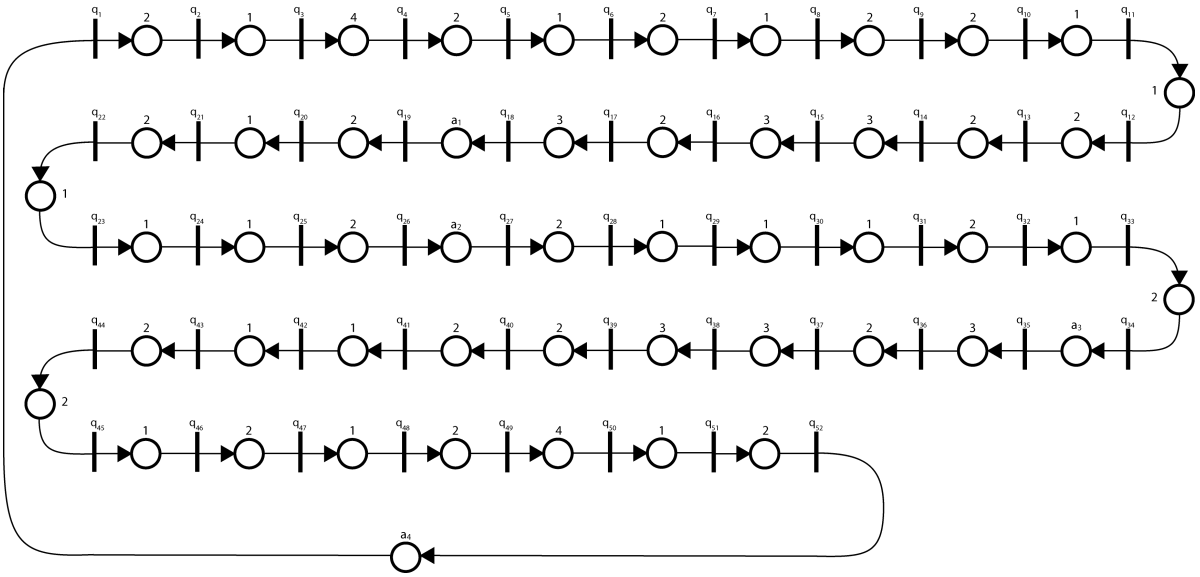
Figuur 2.5: Momentopname op $t = 17$ van de beginvector **v**

Op $t = 17$, het moment dat we kijken, is er net een token gevuurd van q_1. Kijken we naar q_2 en q_3, dan zien we dat de tokens hier al op $t = 2$ zijn gevuurd. Op $t = 17$ bevinden deze zich dan 15 minuten verder in het circuit. Dit zijn respectievelijk de gele en de rode tokens. De tijd dat een token zich in een plaats begeeft is

aangegeven met de kleur van de token onder de plaats. In de praktijk is dit de tijd van hoe lang een tram zich op dat traject bevindt tussen twee haltes. Zo zijn de groene en paarse tokens gevuurd op $t = 1$ en bevinden deze zich op $t = 17$ al 16 minuten op respectievelijk de trajecten tussen $q_3 - q_4$ en $q_4 - q_5$.
Op $t = 0$ is er vanuit q_6 gevuurd. Deze bevindt zich op dit moment 17 minuten verder en dus op het traject tussen q_1 en q_2, ofwel de licht blauwe token.

2.3. Traject tramlijn 19

We hebben in hoofdstuk 2.2 gezien hoe we een max-plus model kunnen opstellen bij een versimpeld traject van tramlijn 19. We gaan dit nu doen voor ons uiteindelijke, grotere model.
De haltes die in figuur 2.3 zijn samengetrokken, geven we nu weer als losse haltes. Dan komen we tot het volgende Petrinet:



Figuur 2.6: Petrinet van tramlijn 19 met alle haltes

Een grotere afbeelding kunt u vinden in appendix A.

Nu is onze verzameling $\mathcal{Q}$ een stuk groter, en telt wel 52 transitites. Ook de verzameling plaatsen $\mathcal{P}$ is uitgebreid en tevens zijn er aantal nieuwe variabelen ingebracht: a_1, a_2, a_3 en a_4. Dit zijn de plaatsen tussen respectievelijk de haltes Delft station aankomst - Delft station vertrek; TU technopolis aankomst - TU Technopolis vertrek; Delft station aankomst - Delft station vertrek; HMC aankomst - HMC vertrek. Omdat het tussen deze transities om wachttijden gaat, kunnen deze aangepast worden. Dit kan handig zijn om de algehele trajectduur te verlengen wanneer dit gunstiger zou zijn. Hierover zult u ook meer lezen in het volgende hoofdstuk.

2.3.1. Opstellen max-plus model toekomstig traject

De verzameling $\mathcal{Q}$ bestaat uit de volgende transities. Deze zijn iets anders genummerd dan in het versimpelde traject!

q_1 := HMC Anthoniushoeve vertrek

q_2 := Leischenhage

q_3 := Weigelia

q_4 := Oosteinde

q_5 := Oude Middenweg

q_6 := Klaverveld

q_7 := Leidschenveen Centrum

q_8 := Lanen

q_9 := Station Ypenburg

q_{10} := Weidevogellaan

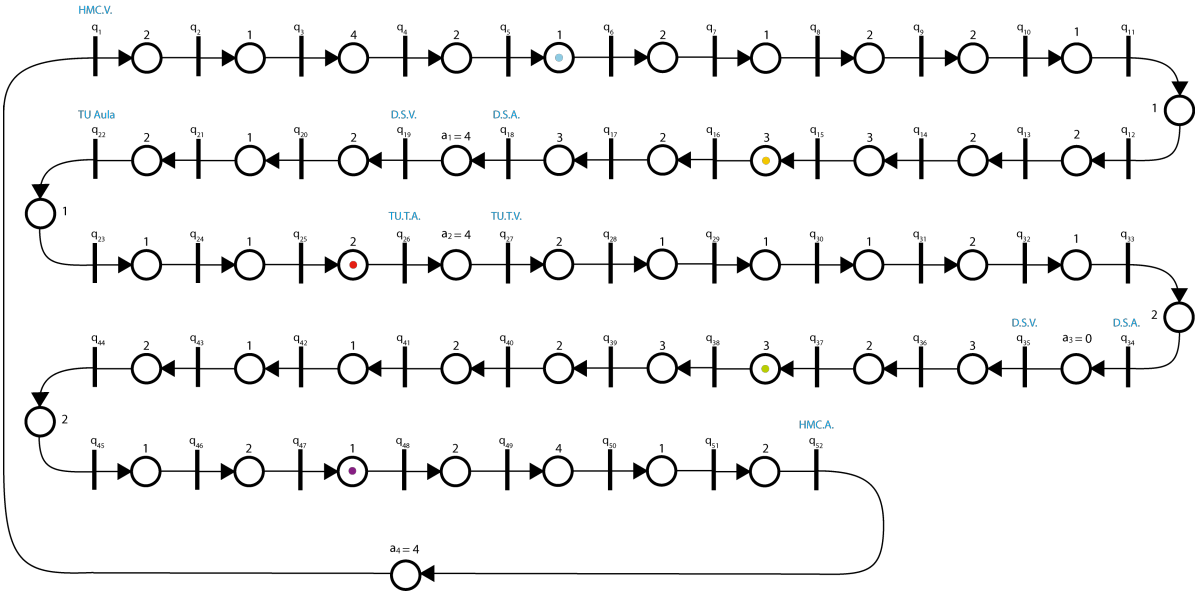
q_{11} := Gruttosingel

q_{12} :=Scholekstersingel

q_{13} := Ypenburg Centrum
q_{14} := Anthony Fokkersingel
q_{15} := Brasserkade
q_{16} := Nieuwe Plantage
q_{17} := Prinsenhof
q_{18} := Delft Station aankomst
q_{19} := Delft station vertrek
q_{20} := Zuidpoort
q_{21} := Mijnbouwplein
q_{22} := TU Aula
q_{23} := TU Mekelpark
q_{24} := TU Sport & Cultuur
q_{25} := TU Kluyverpark
q_{26} := TU Technopolis aankomst
q_{27} := TU Technopolis vertrek
q_{28} := TU Kluyverpark
q_{29} := TU Sport & Cultuur
q_{30} := TU Mekelpark
q_{31} := TU Aula
q_{32} := Mijnbouwplein
q_{33} := Zuidpoort
q_{34} := Delft Station aankomst
q_{35} := Delft Station vertrek
q_{36} := Prinsenhof
q_{37} := Nieuwe Plantage
q_{38} := Brasserskade
q_{39} := Anthony Fokkersingel
q_{40} := Ypenburg Centrum
q_{41} := Scholekstersingel
q_{42} := Gruttosingel
q_{43} := Weidevogellaan
q_{44} := Station Ypenburg
q_{45} := Lanen
q_{46} := Leidschenveen Centrum
q_{47} := Klaverveld
q_{48} := Oude Middenweg
q_{49} := Oosteinde
q_{50} := Weigelia
q_{51} := Leidschenhage
q_{52} := HMC Anthoniushoeve aankomst

De verzameling $\mathscr{P}$ bestaat uit de plaatsen $\mathscr{P} = \{p_1, ..., p_{52}, a_1, a_2, a_3, a_4\}$. De plaatsen geven de trajectduur tussen de haltes $q_i - q_{i+1}$ weer, met $q_{53} = q_1$. De lijst hiervan wordt niet gegeven, omdat de waarden voor p_i gemakkelijk kunnen worden afgelezen uit het Petrinet.
In ons versimpelde traject vonden we $\lambda = 100/5 = 20$. Het is wenselijk om een totale trajecttijd te kiezen die dus goed deelbaar is door 5. Anders krijgen de reizigers te maken met een lastige dienstregeling. Kies je bv. de waarden van a_1, a_2, a_3 en a_4 zó dat de totale trajecttijd op 105 minuten komt te liggen, dan vertrekt gemiddeld elke 21 minuten een tram en dit is onhandig om te onthouden voor een reiziger.
Omdat een totale trajecttijd van 100 minuten in het versimpelde traject goed werkte, kiezen we hier nu weer voor. In het huidige traject hadden de haltes HMC en Delft station al beiden een wachttijd van vier minuten. Bij Delf station was deze tijd nodig om reizigers van de trein te kunnen laten over te stappen op de tram. Op HMC was de wachttijd nodig omdat dit de eind-/beginhalte is en de tram tijd nodig heeft om om te keren en al haar reizigers te laten uitstappen. We kiezen dus voor soortgelijke wachttijden voor de dienstregeling die vanaf 2020 moet gaan gelden. Voor de halte HMC en Delft station (in de richting TU campus) blijven deze vier minuten. De halte TU Technopolis, de eindhalte op de TU campus, krijgt ook een wachttijd van 4 minuten om alle reizigers uit te laten stappen en de tram om te laten keren.
Om een trajecttijd van 100 minuten te behouden kiezen we voor a_1, a_2, a_3 en a_4 de volgende waarden: $a_1 = 4, a_2 = 4, a_3 = 4$ en $a_4 = 0$

Vervolgens plaatsen we onze tokens (trams) willekeurig, maar enigzins gelijkmatige verdeeld over het gehele traject in de plaatsen $p_5, p_{15}, p_{25}, p_{37}$ en p_{47}. We krijgen dan de situatie zoals die te zien is in figuur 2.7. Hierbij zijn een aantal transities aangegeven met de naam van de halte om het een overzichtelijk te houden.



Figuur 2.7: Toekomstige traject tramlijn 19 met vijf trams

We stellen ons max-plus model op d.m.v. de volgende vergelijkingen:

$$
\begin{aligned}
x_1(k+1) &= x_{52}(k+1)+4\\
x_2(k+1) &= x_1(k+1)+2\\
x_3(k+1) &= x_2(k+1)+1\\
x_4(k+1) &= x_3(k+1)+4\\
x_5(k+1) &= x_4(k+1)+2\\
x_6(k+1) &= x_5(k)+1\\
&\vdots\\
x_{16}(k+1) &= x_{15}(k)+3\\
&\vdots\\
x_{26}(k+1) &= x_{25}(k)+2\\
&\vdots\\
x_{38}(k+1) &= x_{37}(k)+3\\
&\vdots\\
x_{48}(k+1) &= x_{47}(k)+3\\
&\vdots\\
x_{52} &= x_{51}(k+1)+2
\end{aligned}
$$

Net zoals bij het versimpelde traject kunnen we vergelijkingen van de vorm $x_i(k+1) = x_{i-1}(k+1)$ omschrijven naar vergelijkingen die afhankelijk zijn van een (k)-term. We vinden dan het volgende stelsel van vergelijkingen:

$$
\begin{aligned}
x_1(k+1) &= x_{47}(k) + 14 & x_2(k+1) &= x_{47}(k) + 16\\
x_3(k+1) &= x_{47}(k) + 17 & x_4(k+1) &= x_{47}(k) + 21\\
x_5(k+1) &= x_{47}(k) + 23 & x_6(k+1) &= x_5(k) + 1\\
x_7(k+1) &= x_5(k) + 3 & x_8(k+1) &= x_5(k) + 4\\
x_9(k+1) &= x_5(k) + 6 & x_{10}(k+1) &= x_5(k) + 8\\
x_{11}(k+1) &= x_5(k) + 9 & x_{12}(k+1) &= x_5(k) + 10\\
x_{13}(k+1) &= x_5(k) + 12 & x_{14}(k+1) &= x_5(k) + 14\\
x_{15}(k+1) &= x_5(k) + 17 & x_{16}(k+1) &= x_{15}(k) + 3\\
x_{17}(k+1) &= x_{15}(k) + 5 & x_{18}(k+1) &= x_{15}(k) + 8\\
x_{19}(k+1) &= x_{15}(k) + 12 & x_{20}(k+1) &= x_{15}(k) + 14\\
x_{21}(k+1) &= x_{15}(k) + 15 & x_{22}(k+1) &= x_{15}(k) + 17\\
x_{23}(k+1) &= x_{15}(k) + 18 & x_{24}(k+1) &= x_{15}(k) + 19\\
x_{25}(k+1) &= x_{15}(k) + 20 & x_{26}(k+1) &= x_{25}(k) + 2\\
x_{27}(k+1) &= x_{25}(k) + 6 & x_{28}(k+1) &= x_{25}(k) + 8\\
x_{29}(k+1) &= x_{25}(k) + 9 & x_{30}(k+1) &= x_{25}(k) + 10\\
x_{31}(k+1) &= x_{25}(k) + 11 & x_{32}(k+1) &= x_{25}(k) + 13\\
x_{33}(k+1) &= x_{25}(k) + 14 & x_{34}(k+1) &= x_{25}(k) + 16\\
x_{35}(k+1) &= x_{25}(k) + 16 & x_{36}(k+1) &= x_{25}(k) + 19\\
x_{37}(k+1) &= x_{25}(k) + 21 & x_{38}(k+1) &= x_{37}(k) + 3\\
x_{39}(k+1) &= x_{37}(k) + 6 & x_{40}(k+1) &= x_{37}(k) + 8\\
x_{41}(k+1) &= x_{37}(k) + 10 & x_{42}(k+1) &= x_{37}(k) + 11\\
x_{43}(k+1) &= x_{37}(k) + 12 & x_{44}(k+1) &= x_{37}(k) + 14\\
x_{45}(k+1) &= x_{37}(k) + 16 & x_{46}(k+1) &= x_{37}(k) + 17\\
x_{47}(k+1) &= x_{37}(k) + 19 & x_{48}(k+1) &= x_{47}(k) + 1\\
x_{49}(k+1) &= x_{47}(k) + 3 & x_{50}(k+1) &= x_{47}(k) + 7\\
x_{51}(k+1) &= x_{47}(k) + 8 & x_{52}(k+1) &= x_{47}(k) + 10
\end{aligned}
$$

Schrijven we dit om naar een matrix stelsel, dan krijgen we een grote 52 x 52 matrix A. Omdat verreweg de meeste elementen van deze matrix gelijk aan ϵ zijn, is de matrix weergeven als een blokmatrix, waarbij de elementen $A_1, ..., A_6$ weer staan voor (kleinere) matrices (in dit geval vectoren).

$$
\begin{array}{c}
 & \{1-4\} & \{5\} & \{6-14\} & \{15\} & \{16-24\} & \{25\} & \{26-36\} & \{37\} & \{38-46\} & \{47\} & \{47-52\}\\
\{1-5\} & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & A_1 & \epsilon\\
\{6-15\} & \vdots & A_2 & & & & & & & & & \vdots\\
\{16-25\} & \vdots & & & A_3 & & & & & & & \vdots\\
\{26-37\} & \vdots & & & & & A_4 & & & & & \vdots\\
\{38-47\} & \vdots & & & & & & & A_5 & & & \vdots\\
\{48-52\} & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & A_6 & \epsilon
\end{array}
$$

met,

$$A_1 = \begin{pmatrix} 14 \\ 16 \\ 17 \\ 21 \\ 23 \end{pmatrix}, A_2 = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 6 \\ 8 \\ 9 \\ 10 \\ 12 \\ 14 \\ 17 \end{pmatrix}, A_3 = \begin{pmatrix} 3 \\ 5 \\ 8 \\ 12 \\ 14 \\ 15 \\ 17 \\ 18 \\ 19 \\ 20 \end{pmatrix}, A_4 = \begin{pmatrix} 2 \\ 6 \\ 8 \\ 9 \\ 10 \\ 11 \\ 13 \\ 14 \\ 16 \\ 16 \\ 19 \\ 21 \end{pmatrix}, A_5 = \begin{pmatrix} 3 \\ 6 \\ 8 \\ 10 \\ 11 \\ 12 \\ 14 \\ 16 \\ 17 \\ 19 \end{pmatrix}, A_6 = \begin{pmatrix} 1 \\ 3 \\ 7 \\ 8 \\ 10 \end{pmatrix}$$

We zien dat zelfs in bloknotatie onze matrix A erg groot blijft. Omdat er eigenlijk maar vijf variabelen ($x_5, x_{15}, x_{25}, x_{37}$ en x_{47}) zijn waarvan de andere variabelen afhangen, kiezen we in het vervolg om een kleinere 5x5 matrix A' te weergeven. We passen hierbij nog steeds de vergelijking 1.8 toe en zo valt het max-plus model in matrixnotatie $\mathbf{x}'(k+1) = A' \otimes \mathbf{x}'(k)$ te schrijven met:

$$\mathbf{x}'(k+1) = \begin{pmatrix} x_5(k+1) \\ x_{15}(k+1) \\ x_{25}(k+1) \\ x_{37}(k+1) \\ x_{47}(k+1) \end{pmatrix}, A' = \begin{pmatrix} \epsilon & \cdots & \cdots & \cdots & 23 \\ 17 & \ddots & & & \epsilon \\ \epsilon & 20 & \ddots & & \vdots \\ \vdots & \ddots & 21 & \ddots & \vdots \\ \epsilon & \cdots & \cdots & 19 & \epsilon \end{pmatrix} \text{ en } \mathbf{x}'(k) = \begin{pmatrix} x_5(k) \\ x_{15}(k) \\ x_{25}(k) \\ x_{37}(k) \\ x_{47}(k) \end{pmatrix} \tag{2.17}$$

De overige waarden van $x_1, \ldots, x_{52}$ kunnen we vinden door de waarden van $x_5, x_{15}, x_{25}, x_{37}, x_{47}$ in ons stelsel van vergelijkingen in te vullen.
In de rest van dit hoofdstuk zullen we steeds de verkorte weergave van $\mathbf{x}'$ gebruiken, maar in Python is natuurlijk gerekend met de gehele vector en valt elke waarde van $x_1, \ldots, x_{52}$ terug te halen.

2.3.2. Power algoritme toegepast traject tramlijn 19

Net zoals bij het versimpelde traject zullen we de stappen van het power algoritme doorlopen om de juiste beginvector te vinden voor ons max-plus model.

1. Neem een willekeurige beginvector $\mathbf{x}(0)$ zodanig dat $\mathbf{x}(0)$ tenminste één niet-eindig element bevat

Als beginvector nemen we weer de vector $\mathbf{x}(0) = \mathbf{0}$

2. Herhaal stap (1.8) totdat er gehele getallen p, q, met $p > q \geq 0$ zijn en een reëel getal c zodanig dat $\mathbf{x}(p) = \mathbf{x}(q) \otimes c$. Er is dan periodiek gedrag bereikt.

We berekenen de vertrektijden van $\mathbf{x}'(k+1)$, voor $k = 0, \ldots, 5$ via (1.8):

$$\mathbf{x}'(1) = \begin{pmatrix} 23 \\ 17 \\ 20 \\ 21 \\ 19 \end{pmatrix}, \mathbf{x}'(2) = \begin{pmatrix} 42 \\ 40 \\ 37 \\ 41 \\ 39 \end{pmatrix}, \mathbf{x}'(3) = \begin{pmatrix} 63 \\ 59 \\ 60 \\ 58 \\ 60 \end{pmatrix}, \mathbf{x}'(4) = \begin{pmatrix} 83 \\ 80 \\ 79 \\ 81 \\ 77 \end{pmatrix}, \mathbf{x}'(5) = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix}, \mathbf{x}'(6) = \begin{pmatrix} 123 \\ 117 \\ 120 \\ 121 \\ 119 \end{pmatrix}, \ldots$$

We zien dat net als in de versimpelde weergave van ons traject, ook hier periodiek gedrag optreedt bij $k = 5$. Er geldt namelijk $\mathbf{x}'(6) = \mathbf{x}'(1) \otimes 100$. Ook voor de gehele vector $\mathbf{x}$ geldt $\mathbf{x}(6) = \mathbf{x}(1) \otimes 100$. Dit kan gecontroleerd worden met Python.

3. Bereken de eigenwaarde λ via $\lambda = c/(p-q)$

Dit geeft ons, net zoals het bij het versimpelde traject, $\lambda = 100/(6-1) = 20$

4. Bereken de eigenvector via $\mathbf{v} = \bigoplus_{j=1}^{p-q} \left(\lambda^{\otimes(p-q-j)} \otimes \mathbf{x}(q+j-1)\right)$

We berekenen de eigenvector $\mathbf{v}$. Om het overzichtelijk te houden zullen we weer alleen de vector $\mathbf{x}'(k+1)$ weergeven, maar in Python is er natuurlijk gewerkt met de originele vector $\mathbf{x}(k+1)$. We weergeven hieronder daarom ook de verkorte versie van $\mathbf{v}$, namelijk $\mathbf{v}' = \begin{pmatrix} v_5 & v_{15} & v_{25} & v_{37} & v_{47} \end{pmatrix}^\top$

$$\mathbf{v}' = \bigoplus_{j=1}^{p-q} \left(\lambda^{\otimes(p-q-j)} \otimes \mathbf{x}'(q+j-1)\right) \tag{2.18}$$

$$= \bigoplus_{j=1}^{5} \left(\lambda^{\otimes(5-j)} \otimes \mathbf{x}'(j)\right) \tag{2.19}$$

$$= 20^{\otimes 4} \otimes \mathbf{x}'(1) \oplus 20^{\otimes 3} \otimes \mathbf{x}'(2) \oplus 20^{\otimes 2} \otimes \mathbf{x}'(3) \oplus 20 \otimes \mathbf{x}'(4) \oplus \mathbf{x}'(5) \tag{2.20}$$

$$= 80 \otimes \begin{pmatrix} 23 \\ 17 \\ 20 \\ 21 \\ 19 \end{pmatrix} \oplus 60 \otimes \begin{pmatrix} 42 \\ 40 \\ 37 \\ 41 \\ 39 \end{pmatrix} \oplus 40 \otimes \begin{pmatrix} 63 \\ 59 \\ 60 \\ 58 \\ 60 \end{pmatrix} \oplus 20 \otimes \begin{pmatrix} 83 \\ 80 \\ 79 \\ 81 \\ 77 \end{pmatrix} \oplus \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} \tag{2.21}$$

$$= \begin{pmatrix} 103 \\ 97 \\ 100 \\ 101 \\ 99 \end{pmatrix} \oplus \begin{pmatrix} 102 \\ 100 \\ 97 \\ 101 \\ 99 \end{pmatrix} \oplus \begin{pmatrix} 103 \\ 99 \\ 100 \\ 98 \\ 100 \end{pmatrix} \oplus \begin{pmatrix} 103 \\ 100 \\ 99 \\ 101 \\ 97 \end{pmatrix} \oplus \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} \tag{2.22}$$

$$= \begin{pmatrix} 103 \\ 100 \\ 100 \\ 101 \\ 100 \end{pmatrix} := \mathbf{v}' \tag{2.23}$$

Omdat de beginvector niet uniek is, kunnen we deze herschrijven: $\mathbf{v}' = \begin{pmatrix} 22 & 19 & 19 & 20 & 19 \end{pmatrix}^\top$.
Passen we nu de vergelijking $\mathbf{x}(k+1) = A \otimes \mathbf{x}(k)$, toe, met als beginvector $\mathbf{x}(0) = \mathbf{v}$, dan vinden we de volgende vertrektijden. Ook hierbij weergeven we weer de verkorte versie $\mathbf{x}'$ weer.

$$\mathbf{x}(1) = \begin{pmatrix} 42 \\ 39 \\ 39 \\ 40 \\ 39 \end{pmatrix}, \mathbf{x}(2) = \begin{pmatrix} 62 \\ 59 \\ 59 \\ 60 \\ 59 \end{pmatrix}, \mathbf{x}(3) = \begin{pmatrix} 82 \\ 79 \\ 79 \\ 80 \\ 79 \end{pmatrix}, \mathbf{x}(4) = \begin{pmatrix} 102 \\ 99 \\ 99 \\ 100 \\ 99 \end{pmatrix}, \mathbf{x}(5) = \begin{pmatrix} 122 \\ 119 \\ 119 \\ 120 \\ 119 \end{pmatrix}, \mathbf{x}(6) = \begin{pmatrix} 142 \\ 139 \\ 139 \\ 140 \\ 139 \end{pmatrix}, \cdots \tag{2.24}$$

We zien in (2.24) dat we vanaf de eerste stap al een regelmatige dienstregeling krijgen, zoals gewenst.

2.4. Conclusie hoofdstuk 2

In dit hoofdstuk hebben we een geschikte beginvector gevonden voor de dienstregeling van tramlijn 19 met vijf trams. Eerst is er met een versimpeld model gewerkt waarin haltes zijn samengevoegd tot dummihaltes. Daarna hebben we het model uitgebreid en vertrektijden gevonden voor alle haltes.
In het volgende hoofdstuk zullen we nog gaan kijken naar de overgang tussen een spits-en daluur. Voor de spitsuren stellen we een dienstregeling met zeven trams op.

3

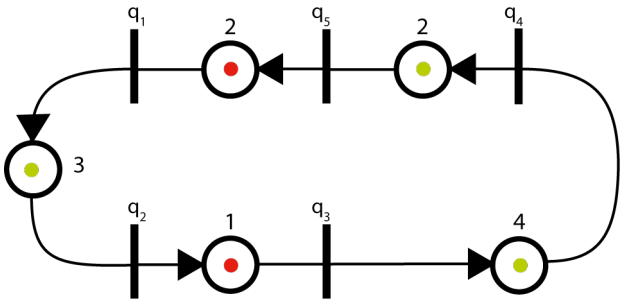
Overgang spits- naar daluur

3.1. Inleiding probleem

In hoofdstuk 2 hebben we een dienstregeling voor tramlijn 19 opgesteld met vijf trams. Deze rijden elke 20 minuten. Echter, in het huidige traject van tramlijn 19, dus zonder de TU haltes, geldt een andere dienstregeling. Hier rijdt elke 15 minuten een tram tot 20 u 's avonds. Hierna rijdt er elke 20 minuten een tram.
Er is een extra spoor aangelegd om o.a. in de spitsuren extra trams te laten rijden [7]. Omdat het traject een flink stuk langer is geworden en we ervan uitgaan dat de dienstregeling van 15 minuten wenselijk is om overdag aan te houden, gaan we er in dit hoofdstuk vanuit dat HTM extra trams zal aanschaffen om dit te verwezenlijken.
De totale trajectijd van tramlijn 19 met uitbreiding naar de TU haltes is zonder de wachttijden op HMC, Delft station en TU Technopolis 88 minuten. In de huidige dienstregeling is er echter een wachtttijd van vier minuten op HMC en op Delft station (in de richting HMC - Delft station). Zoals al eerder werd genoemd kunnen we hieruit concluderen dat het belangrijk is dat de begin- en eindhaltes een wachttijd van minimaal vier minuten hebben. Ook moet Delft station in de richting HMC-TU technopolis een wachttijd van minimaal vier minuten hebben om reizigers de tijd te geven over te stappen van de trein op tramlijn 19.
Dit maakt dat onze trajecttijd dus minimaal 88 + 4 + 4 + 4 = 100 minuten lang is. Een dienstregeling van 15 minuten met zes trams is dus al uitgesloten: $6 \times 15 = 90$ minuten en dit is minder lang dan de toekomstige trajectduur. Aangezien trams duur zijn om aan te schaffen, willen we niet meer trams aanschaffen dan nodig is. Zeven trams blijkt voor de dienstregeling van 15 minuten wel geschikt. Het is dan wel nodig om ergens het traject een wachttijd van vijf minuten toe te voegen, aangezien $7 \times 15 = 105$ minuten.
In dit hoofdstuk beginnen we dus met het opstellen van een model voor de spitsuren. Hierna zoeken we een manier om de dienstregeling van zeven trams over te laten gaan in een dienstregeling van vijf trams. We beginnen weer met een versimpeld, kleiner traject.Hierna zullen we het toekomstige traject van tramlijn 19 bekijken.

3.2. Versimpeld traject

In dit hoofdstuk bekijken we een klein traject van twaalf minuten met twee óf drie trams. De verzameling transities bestaat uit $\mathcal{Q} = q_1, ..., q_5$, waarbij q_3 de rol vervult van een remise: hier worden de trams gestald wanneer ze buiten dienst zijn. Als de dienstregeling van drie naar twee trams gaat, zal de derde tram dus via q_3 het traject verlaten.
Het Petrinetwerk behorende bij dit traject volgt hieronder:



Figuur 3.1: Petrinet van een klein traject met twee of drie trams

In dit Petrinet is de situatie met twee trams aangegeven met rode tokens, en de situatie met drie trams met groene tokens.

Aan de hand van dit Petrinet stellen weer een max plus model op. Hierbij krijgen we de volgende vergelijkingen:

Twee trams

$$
\begin{aligned}
x_1(k+1) &= x_5(k)+2\\
x_2(k+1) &= x_5(k)+5\\
x_3(k+1) &= x_2(k)+1\\
x_4(k+1) &= x_2(k)+5\\
x_5(k+1) &= x_2(k)+7
\end{aligned}
$$

In matrix notatie geeft dit:

$$
\mathbf{x}(k+1) = \otimes\mathbf{x}(k) = \begin{pmatrix} \epsilon & \cdots & \cdots & \cdots & 2 \\ \vdots & \ddots & & & 5 \\ \vdots & 1 & \ddots & & \epsilon \\ \vdots & 5 & & \ddots & \vdots \\ \epsilon & 7 & \epsilon & \cdots & \epsilon \end{pmatrix} \otimes \mathbf{x}(k) \tag{3.1}
$$

Via het power algoritme vinden we de beginvector $\mathbf{x}(0) = (2,5,0,4,6)^\top$ en de eigenwaarde $\lambda = 6$. Deze geeft de volgende vertrektijden voor $k = 0,1,2,\ldots$:

$$
\mathbf{x}(1) = \begin{pmatrix} 8 \\ 11 \\ 6 \\ 10 \\ 12 \end{pmatrix}, \mathbf{x}(2) = \begin{pmatrix} 14 \\ 17 \\ 12 \\ 16 \\ 18 \end{pmatrix}, \mathbf{x}(3) = \begin{pmatrix} 20 \\ 23 \\ 18 \\ 22 \\ 24 \end{pmatrix}, \mathbf{x}(4) = \begin{pmatrix} 26 \\ 29 \\ 24 \\ 28 \\ 30 \end{pmatrix}, \cdots \tag{3.2}
$$

Elke zes minuten vertrekt er dus een tram.

Drie trams

$$
\begin{aligned}
y_1(k+1) &= y_4(k)+4 && (3.3)\\
y_2(k+1) &= y_1(k)+3 && (3.4)\\
y_3(k+1) &= y_1(k)+4 && (3.5)\\
y_4(k+1) &= y_3(k)+4 && (3.6)\\
y_5(k+1) &= y_4(k)+2 && (3.7)
\end{aligned}
$$

In matrix notatie geeft dit:

$$\mathbf{y}(k+1) = B \otimes \mathbf{y}(k) = \begin{pmatrix} \epsilon & \cdots & \cdots & 4 & \epsilon \\ 3 & \ddots & & & \vdots \\ 4 & & \ddots & & \vdots \\ \epsilon & & 4 & \ddots & \vdots \\ \epsilon & \cdots & \cdots & 2 & \epsilon \end{pmatrix} \otimes \mathbf{y}(k) \tag{3.8}$$

In dit geval geeft de beginvector $\mathbf{y}(0) = \mathbf{0}$ ons direct een regelmatige dienstregeling. Met $\lambda = 4$ en $k = 1, 2, \ldots$ vinden we:

$$\mathbf{y}(1) = \begin{pmatrix} 4 \\ 3 \\ 4 \\ 4 \\ 2 \end{pmatrix}, \quad \mathbf{y}(2) = \begin{pmatrix} 6 \\ 8 \\ 8 \\ 7 \\ 8 \end{pmatrix}, \quad \mathbf{y}(3) = \begin{pmatrix} 10 \\ 12 \\ 12 \\ 11 \\ 12 \end{pmatrix}, \quad \mathbf{y}(4) = \begin{pmatrix} 14 \\ 16 \\ 16 \\ 15 \\ 16 \end{pmatrix}, \quad \cdots \tag{3.9}$$

3.2.1. Overgang drie naar twee trams

Stel we beginnen met drie trams die elke vier minuten rijden. Na een uur willen we graag overgaan op een daluren dienstregeling, met maar twee trams die elke zes minuten rijden. Halte q_3 fungeert hier als remise. Via q_3 verlaat de derde tram het traject. Ook zouden we graag willen dat als het eerste uur voorbij is, we z.s.m. via de dienstregeling van zes minuten gaan rijden.
We kijken naar het tijdstip $t = 60$. De dienstregeling met drie trams bereikt dit tijdstip op $k = 15$ met $\mathbf{y}(15) = \begin{pmatrix} 60 & 59 & 60 & 60 & 58 \end{pmatrix}^\top$. Rond hetzelfde tijdstip is de dienstregeling van twee trams nog in de negende stap, met $\mathbf{x}(9) = \begin{pmatrix} 56 & 59 & 54 & 58 & 60 \end{pmatrix}^\top$. We willen dat de dienstregeling van drie trams overvloeit in die van twee trams. Omdat we met max-plus algebra werken, kunnen we dit alleen doen door het model van drie trams te vertragen. Versnellen is in de max plus algebra niet mogelijk binnen dit model. Bovendien staan de trajecttijden vast en kunnen deze niet korter worden door trams harder te laten rijden (zie hoofdstuk 2.2).
De vertrektijden van $\mathbf{x}(9)$ zijn niet allemaal later dan de vertrektijden van $\mathbf{y}(15)$. Overgaan van $\mathbf{y}(15)$ naar $\mathbf{x}(9)$ zou dus betekenen dat we ergens ons model moeten versnellen en dit is niet mogelijk. We kijken daarom naar de vector $\mathbf{x}(10) = \begin{pmatrix} 62 & 65 & 60 & 64 & 66 \end{pmatrix}^\top$. De vertrektijden hiervan liggen allemaal hoger dan die van $\mathbf{y}(15)$ en is dus geschikt om als volgende vector te kiezen in de dienstregeling.
Hieronder volgt samengevat de situatie rond $t = 60$ en een afbeelding van de overgang.

Situatieschets rond t=60

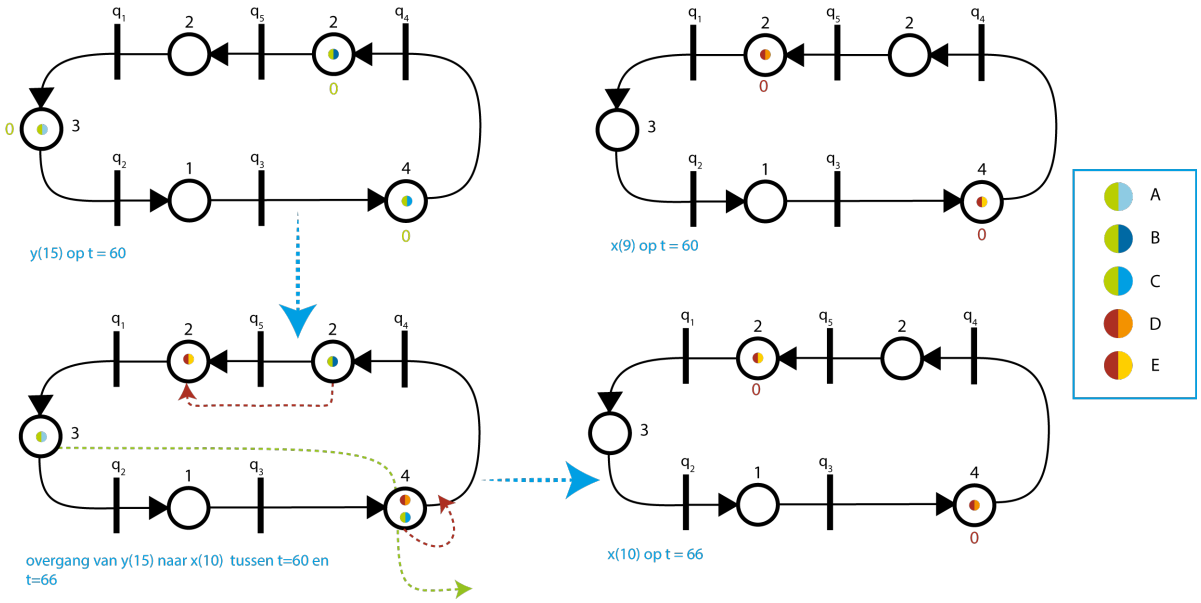
- Op $t = 0$ begint de dienstregeling met drie trams die elke *vier* minuten rijden. Na $k = 15$ schakelt op $t = 60$ de dienstregeling van drie trams over op een dienstregeling van twee trams die elke *zes* minuten rijden.

- De eerstvolgende vertrektijden die bij de dienstregeling van twee trams mogelijk zijn, zijn die van $\mathbf{x}(10)$. De dienstregeling van drie trams moet dus in de volgende stap vertraagd worden tot een dienstregeling van twee trams.

- Er rijdt één tram via q_3 naar de remise.

Het overbrengen van de dienstregeling van drie naar twee trams doen we met een *overgangsmatrix* die we O noemen.

We lossen hiervoor de vergelijking $O \otimes \mathbf{y}(15) = \mathbf{x}(10)$ op. Er zijn verschillende matrices die hiervoor geschikt zijn, maar in dit geval is gekozen voor de diagonaalmatrix O die ons geeft:



Figuur 3.2: Situatieweergave van $\mathbf{y}(15)$ en $\mathbf{x}(10)$ en overgangsweergave tussen t=60 en t=66

$$\begin{pmatrix} 2 & \epsilon & \cdots & \cdots & \epsilon \\ \epsilon & 6 & \ddots & & \vdots \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \vdots & & \ddots & 4 & \epsilon \\ \epsilon & \cdots & \cdots & \epsilon & 8 \end{pmatrix} \otimes \mathbf{y}(15) = \mathbf{x}(10) \tag{3.10}$$

Hierbij staan de punten weer voor ϵ.

In figuur 3.2 kunt u zien hoe deze overgang in zijn werking gaat. Een grotere afbeelding is te vinden in appendix A.

In de overgang tussen $\mathbf{y}(15)$ en $\mathbf{x}(10)$ gebeurt het volgende:

- Op t=60 staan trams A, B en C zoals linksboven. We moeten op t = 66 eindigen in de situatie van rechtsonder, waarbij we alleen nog tram D en E in het circuit hebben
- $A \rightarrow$ remise: op t=60 bevindt A zich op het traject tussen q_1 en q_2. Op t=63 vertrekt A vanaf q_2. Op t=64 komt A aan op q_3 en vertrekt vanuit hier naar de remise. Op t=66 bevindt tram A zich niet meer in het traject en staat op de remise.
- $C \rightarrow D$: op t=60 bevindt tram D zich op halte q_3. Hier blijft hij zes minuten wachten, om vervolgens als tram D op t=66 te vertrekken vanuit q_3 richting q_4
- $B \rightarrow E$: op t=60 bevindt tram B zich op het traject tussen q_4 en q_5. Op t = 62 komt tram B aan op q_5, waarna hij vier minuten wacht en op $t = 66$ als tram E verder vertrekt richting q_1.

We zien dus dat er vertraagd wordt om uiteindelijk over te gaan op de nieuwe dienstregeling. Nadat we de overgangsmatrix O hebben gebruikt om tot $\mathbf{x}(10)$ te komen, zal A weer fungeren om de volgende vertrektijden te berekenen via $\mathbf{x}(k+1) = \mathbf{A} \otimes \mathbf{x}(k)$

3.2.2. Conclusie overgang versimpeld traject

In het versimpelde traject volgen we voor één uur de dienstregeling voor drie trams. Hierna zijn we over gegaan op een dienstregeling voor twee trams. Hierbij hebben we gebruik gemaakt van een overgangsmatrix O. De vertrektijden van de haltes $q_1, \ldots, q_5$ worden dan:

$$\mathbf{x_3}(1) = \begin{pmatrix}4\\3\\4\\4\\2\end{pmatrix}, \quad \mathbf{x}(2) = \begin{pmatrix}6\\8\\8\\7\\8\end{pmatrix}, \quad \ldots, \quad \mathbf{x_3}(15) = \begin{pmatrix}60\\59\\60\\60\\58\end{pmatrix}, \quad \mathbf{x_2}(10) = \begin{pmatrix}62\\65\\60\\64\\66\end{pmatrix}, \quad \mathbf{x_2}(11) = \begin{pmatrix}68\\71\\66\\70\\72\end{pmatrix}, \quad \ldots \tag{3.11}$$

De code die in dit hoofdstuk is gebruikt om de vertrektijden te berekenen kunt u terug vinden in appendix B.

3.3. Tramlijn 19

In hoofdstuk 2.3 hebben we al gezien hoe we een dienstregeling voor vijf trams opstellen. In hoofdstuk 3.1 is onderbuowd waarom het nodig is twee extra trams aan te schaffen voor de spitsuren. Ook is er genoemd dat er een spitsuur van 's ochtends tot 's avonds 20 uur geldt. Hierbij rijdt elke 15 minuten een tram. Hierna beginnen de daluren met elke 20 minuten een tram. Een dienstregeling hiervoor is al deels opgesteld in hoofdstuk 2.
In dit hoofdstuk zullen we een dienstregeling voor zeven trams opstellen. We zullen een overgangsmatrix O bepalen voor het tijdstip ± 20 uur. Daarnaast zijn er nog twee voorwaarden waarmee rekening gehouden moet worden:

- De eerste tram van de dag vertrekt rond 5:30 's ochtends vanaf HMC Anthoniushoeve richting TU Technopolis

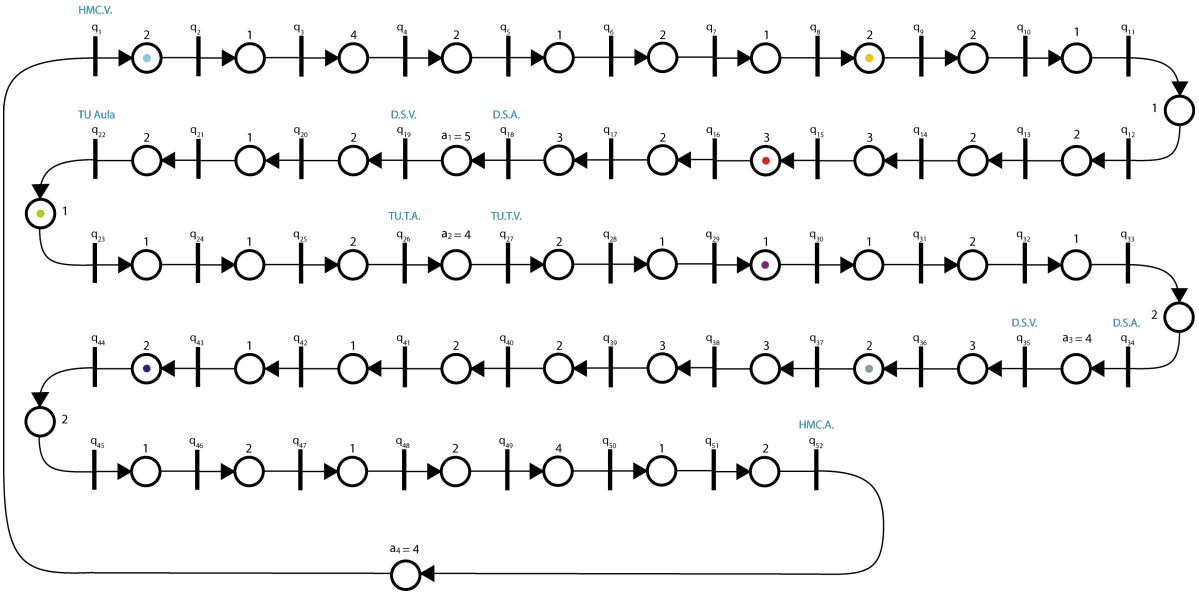
- Om ±08:25 en ±8:40 moet de halte TU Aula worden aangedaan

In de huidige dienstregeling beginnen de trams rond 5:30 te rijden en deze voorwaarde hebben wij dan ook meegenomen in ons onderzoek. 's Nachts is er waarschijnlijk niet of maar een kleine behoefte aan openbaar vervoer. Het is dan erg duur om trams te laten rijden voor enkele reizigers. Daarnaast is het voor de trambestuurders onwenselijk om vroeger dan 5:30 te beginnen met werken.
De tweede voorwaarde heeft te maken met de collegetijden van de TU Delft. De meeste faculteiten liggen bij de halte TU Aula en zijn maximaal vier minuten lopen. De eerste colleges beginnen om 8:45. Als de trams om 8:25 en 8:40 aankomen zijn er twee gunstige shifts voor studenten om met het openbaar vervoer aan te komen op de halte TU Aula. De faculteiten die verder weg liggen, zoals Lucht & Ruimtevaart of TNW-zuid, worden nog op tijd aangedaan door de tram die om 8:25 weer vanaf TU Aula vertrekt.
In het huidige traject dient de halte HMC Anthoniushoeve als remise. Wij zullen dit ook aanhouden in ons model voor het toekomstige traject.

3.3.1. Opstellen dienstregeling zeven trams

We willen dat elke 15 minuten een tram vertekt. Daarom passen we onze wachttijden a_1, a_2, a_3, a_4 dus zodanig aan dat we een totale trajecttijd van 105 (=7 × 15) minuten krijgen.

Hierbij hoort het volgende Petrinet (dit is een aanpassing op het Petrinet uit afbeelding 2.6):



Figuur 3.3: Petrinetwerk zeven trams

De wachttijden op de stations TU Technopolis, Delft station en HMC Anthoniushoeve zijn deels aangepast naar $a_1 = 5, a_2 = 4, a_3 = 4, a_4 = 4$.
Ons max-plus model wordt hierdoor ook wat anders. We plaatsen zeven trams op willekeurige trajecten, maar wel regelmatig verdeeld over het traject. We plaatsen deze in de plaatsen tussen de transities $q_1 - q_2; q_8 - q_9; q_{15} - q_{16}; q_{22} - q_{23}; q_{29} - q_{30}; q_{36} - q_{37}$ en $q_{43} - q_{44}$ (zie ook figuur 3.3). Hierbij horen de volgende vergelijkingen:

$$
\begin{aligned}
y_1(k+1) &= y_{43}(k) + 21 & y_2(k+1) &= y_1(k) + 2\\
y_3(k+1) &= y_1(k) + 3 & y_4(k+1) &= y_1(k) + 7\\
y_5(k+1) &= y_1(k) + 9 & y_6(k+1) &= y_1(k) + 10\\
y_7(k+1) &= y_1(k) + 12 & y_8(k+1) &= y_1(k) + 13\\
y_9(k+1) &= y_8(k) + 2 & y_{10}(k+1) &= y_8(k) + 4\\
y_{11}(k+1) &= y_8(k) + 5 & y_{12}(k+1) &= y_8(k) + 6\\
y_{13}(k+1) &= y_8(k) + 8 & y_{14}(k+1) &= y_8(k) + 10\\
y_{15}(k+1) &= y_8(k) + 13 & y_{16}(k+1) &= y_{15}(k) + 3\\
y_{17}(k+1) &= y_{15}(k) + 5 & y_{18}(k+1) &= y_{15}(k) + 8\\
y_{19}(k+1) &= y_{15}(k) + 12 & y_{20}(k+1) &= y_{15}(k) + 14\\
y_{21}(k+1) &= y_{15}(k) + 15 & y_{22}(k+1) &= y_{15}(k) + 18\\
y_{23}(k+1) &= y_{22}(k) + 1 & y_{24}(k+1) &= y_{22}(k) + 2\\
y_{25}(k+1) &= y_{22}(k) + 3 & y_{26}(k+1) &= y_{22}(k) + 5\\
y_{27}(k+1) &= y_{22}(k) + 9 & y_{28}(k+1) &= y_{22}(k) + 11\\
y_{29}(k+1) &= y_{22}(k) + 12 & y_{30}(k+1) &= y_{29}(k) + 1\\
y_{31}(k+1) &= y_{29}(k) + 2 & y_{32}(k+1) &= y_{29}(k) + 4\\
y_{33}(k+1) &= y_{29}(k) + 5 & y_{34}(k+1) &= y_{29}(k) + 7\\
y_{35}(k+1) &= y_{29}(k) + 11 & y_{36}(k+1) &= y_{29}(k) + 14\\
y_{37}(k+1) &= y_{36}(k) + 2 & y_{38}(k+1) &= y_{36}(k) + 5\\
y_{39}(k+1) &= y_{36}(k) + 8 & y_{40}(k+1) &= y_{36}(k) + 10\\
y_{41}(k+1) &= y_{36}(k) + 12 & y_{42}(k+1) &= y_{36}(k) + 13\\
y_{43}(k+1) &= y_{36}(k) + 14 & y_{44}(k+1) &= y_{43}(k) + 2
\end{aligned}
$$

$$
\begin{array}{lcllcl}
y_{45}(k+1) & = & y_{43}(k) + 4 & y_{46}(k+1) & = & y_{43}(k) + 5 \\
y_{47}(k+1) & = & y_{43}(k) + 7 & y_{48}(k+1) & = & y_{43}(k) + 8 \\
y_{49}(k+1) & = & y_{43}(k) + 10 & y_{50}(k+1) & = & y_{43}(k) + 14 \\
y_{51}(k+1) & = & y_{43}(k) + 15 & y_{52}(k+1) & = & y_{43}(k) + 17
\end{array}
$$

We noteren in het vervolg de gehele vector van vertrektijden in ons model met zeven trams met $\mathbf{y}$. De bijbehorende matrix geven we aan met B. In matrixnotatie geeft dit ons $\mathbf{y}(k+1) = B \otimes \mathbf{y}(k)$.
Omdat alle variabelen afhangen van $y_1, y_8, y_{15}, y_{22}, y_{29}, y_{36}$ en y_{43}, gebruiken we net als in hoofdstuk 2.3.1 een verkorte weergave van $\mathbf{y}$ en B. We noemen deze $\mathbf{y}'$en B' en vinden dan de vergelijking $\mathbf{y}'(k+1) = B' \otimes \mathbf{y}'(k)$, met:

$$
\mathbf{y}'(k+1) = \begin{pmatrix} y_1(k+1) \\ y_8(k+1) \\ y_{15}(k+1) \\ y_{22}(k+1) \\ y_{29}(k+1) \\ y_{36}(k+1) \\ y_{43}(k+1) \end{pmatrix}, B' = \begin{pmatrix} \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & 21 \\ 13 & \ddots & & & & & \epsilon \\ \epsilon & 13 & \ddots & & & & \vdots \\ \vdots & \ddots & 18 & \ddots & & & \vdots \\ \vdots & & \ddots & 12 & \ddots & & \vdots \\ \vdots & & & \ddots & 14 & \ddots & \vdots \\ \epsilon & \cdots & \cdots & \cdots & \epsilon & 14 & \epsilon \end{pmatrix} \text{ en } \mathbf{y}'(k) = \begin{pmatrix} y_1(k) \\ y_8(k) \\ y_{15}(k) \\ y_{22}(k) \\ y_{29}(k) \\ y_{36}(k) \\ y_{43}(k) \end{pmatrix} \tag{3.12}
$$

Op dezelfde manier als bij de dienstregeling voor vijf trams gebruiken we het power algoritme (zie hoofdstuk 2.3.2).
Nemen we de beginvector $\mathbf{y}(0) = \mathbf{0}$, dan treedt er na zeven iteraties periodiek gedrag op. Ofwel: $\mathbf{y}(1) \otimes 105 = \mathbf{y}(8)$. Dit geeft ons zoals verwacht een eigenwaarde van $\lambda = 15$.

We berekenen de eigenvector $\mathbf{w}$. We zullen hier weer slechts enkele elementen van weergeven. Deze verkorte weergave noemen we $\mathbf{w}'$ met:

$$
\mathbf{w}' = \begin{pmatrix} w_1 \\ w_8 \\ w_{15} \\ w_{22} \\ w_{29} \\ w_{36} \\ w_{43} \end{pmatrix} = \begin{pmatrix} 111 \\ 109 \\ 107 \\ 110 \\ 107 \\ 106 \\ 105 \end{pmatrix} = \begin{pmatrix} 19 \\ 17 \\ 15 \\ 18 \\ 15 \\ 14 \\ 13 \end{pmatrix} \tag{3.13}
$$

Waarbij we in de laatste stap $\mathbf{w}$ hebben herschreven zodanig dat het kleinste element van de vector $\mathbf{w}$ overal vanaf getrokken wordt (met de reguliere - operatie).
We nemen nu als beginvector $\mathbf{y}(0) = \mathbf{w}$ en vinden met $\mathbf{y}(k+1) = B \otimes \mathbf{y}(k)$, en voor $k = 1, 2, \ldots$ de vertrektijden:

$$
\mathbf{y}'(1) = \begin{pmatrix} 34 \\ 32 \\ 30 \\ 33 \\ 30 \\ 29 \\ 28 \end{pmatrix}, \mathbf{y}'(2) \begin{pmatrix} 49 \\ 47 \\ 45 \\ 48 \\ 45 \\ 44 \\ 43 \end{pmatrix}, \mathbf{y}'(3) \begin{pmatrix} 64 \\ 62 \\ 60 \\ 63 \\ 60 \\ 59 \\ 58 \end{pmatrix}, \ldots \tag{3.14}
$$

Hierbij zien we dat er direct een regelmatige dienstregeling van 15 minuten volgt.

3.3.2. Voorwaarden dienstregeling spitsuur

Zoals eerder besproken zijn er naast de overgang nog een tweetal voorwaarden die belangrijk zijn bij de spitsdienstregeling met zeven trams. We moeten hier eventueel onze beginvector $\mathbf{y}(0)$ voor aanpassen.

Gunstige vertrektijden TU Aula

Een van de eisen die we aan het begin van dit hoofdstuk stelden, was dat er om ± 8:25 en ± 8:40 een tram bij de halte TU Aula (q_{22}) vanaf station Delft moet aankomen.
Met de beginvector die we nu hebben gekozen, vinden we rond het tijdstip 8:30 de volgende waarden voor $\mathbf{y_{22}}$:

$$y_{22}(12) = 8:18 \quad ; y_{22}(13) = 8:33; \quad y_{22}(14) = 8:48 \tag{3.15}$$

Dit zijn dus nog niet de gewenste vertrijktijden van ± 8:25 en ± 8:40.
We verhogen nu onze beginvector $\mathbf{y}(0)$ met 7. Dit kan, omdat de eigenvector niet uniek is. Dan krijgen we :

$$\mathbf{y}'(0) = \begin{pmatrix} 26 \\ 24 \\ 22 \\ 25 \\ 22 \\ 21 \\ 20 \end{pmatrix} \qquad y_{22}(12) = 8:25; \quad y_{22}(13) = 8:40 \tag{3.16}$$

Dit zijn wel de gewenste vertrektijden vanaf de halte TU Aula. In het vervolg zullen we dan ook deze vector als de beginvector $\mathbf{y}(0)$ kiezen voor de dienstregeling van zeven trams.

Begin- en eindtijd dienstregeling

Een andere voorwaarde was dat de eerste tram rond 5:30 moet beginnen. Het max-plus model zit zodanig in elkaar dat het vanaf $k = 1$ al de vertrektijden worden gegeven als er al *zeven trams* rijden. In de praktijk zal er vanaf 5:30 elk kwartier een tram het traject ingaan vanaf de remise, totdat alle zeven trams zich op het traject bevinden.
We weten dat rond 5:30 de eerste tram vanaf HMC vertrekt. ± 90 minuten later bevinden alle zeven trams zich op het traject. We zien dat onze beginvector $y_1(0)$ gelijk is aan 26. 5:26 is een geschikte begintijd voor de eerste tram. Dit betekent dat vanaf $k = 6$ alle trams zich op het traject bevinden. Vanaf dan kunnen we spreken van een "volledige dienstregeling", omdat dan pas alle trams rijden.
Een andere voorwaarde is dat de laatste tram rond 01:00 bij HMC moet aankomen. In de huidige dienstregeling rijdt er na dit tijdstip geen tram meer en in de toekomstige dienstregeling zullen wij dit dan ook aanhouden. Rond 1:00 uur rijdt dus de laatste tram via HMC naar de remise. Dit gebeurt op $k = 59$ in de dienstregeling van 5 trams. We vinden dan de waarde $x_{52}(59) = 00:49$. Om 00:49 zal dus de laatste tram naar de remise vertrekken en vanaf dan zullen er geen trams meer op het traject rijden. Dat betekent dat vanaf $k = 60$ de dienstregeling volledig gestopt is en alle trams uit het traject zijn.

3.3.3. Overgang spits- naar daluur

Het laatste wat nog gedaan moet worden is een overgang bepalen van het spitsuur naar het daluur. Rond ± 20 uur moeten er twee trams die naar HMC Anthoniushoeve vanuit Delft rijden, naar de remise.
In de dienstregeling voor zeven trams liggen voor $k = 59$ bijna alle vertrektijden rond 20:00. Bij de dienstregeling van vijf trams gebeurt dit voor $k = 45$. De waarden van $\mathbf{x}(45), \mathbf{y}(58), \mathbf{y}(59)$ worden hieronder in verkorte weergave weergeven (dit zijn dezelfde elementen uit $\mathbf{x}$ en $\mathbf{y}$ als in de hoofdstukken 2.3.1 en 3.3.1). De vertrektijden voor zeven trams zijn aan de hand berekend van de met 7 verhoogde beginvector

$$\mathbf{x}' = \begin{pmatrix} x_5 \\ x_{15} \\ x_{25} \\ x_{37} \\ x_{47} \end{pmatrix}, \mathbf{x}'(45) = \begin{pmatrix} 20:22 \\ 20:19 \\ 20:19 \\ 20:20 \\ 20:19 \end{pmatrix}, \quad ; \quad \mathbf{y}' = \begin{pmatrix} y_1 \\ y_8 \\ y_{15} \\ y_{22} \\ y_{29} \\ y_{36} \\ y_{43} \end{pmatrix}, \mathbf{y}'(58) = \begin{pmatrix} 19:56 \\ 19:54 \\ 19:52 \\ 19:55 \\ 19:52 \\ 19:51 \\ 19:50 \end{pmatrix}, \mathbf{y}'(59) = \begin{pmatrix} 20:11 \\ 20:09 \\ 20:07 \\ 20:10 \\ 20:07 \\ 20:06 \\ 20:05 \end{pmatrix} \tag{3.17}$$

We willen nu dus van de vector $\mathbf{y}(58)$ overgaan naar $\mathbf{x}(45)$. We kiezen juist deze vectoren rond het tijdstip 20 uur, omdat elk element uit $\mathbf{y}(58)$ kleiner of gelijk is aan de elementen van $\mathbf{x}(45)$. We zagen al eerder dat dit

nodig is, omdat we in een max-plus model alleen kunnen vertragen. Om die reden moeten in een volgende stap de vertrektijden dus later liggen.

Opstellen overgangsmatrix O

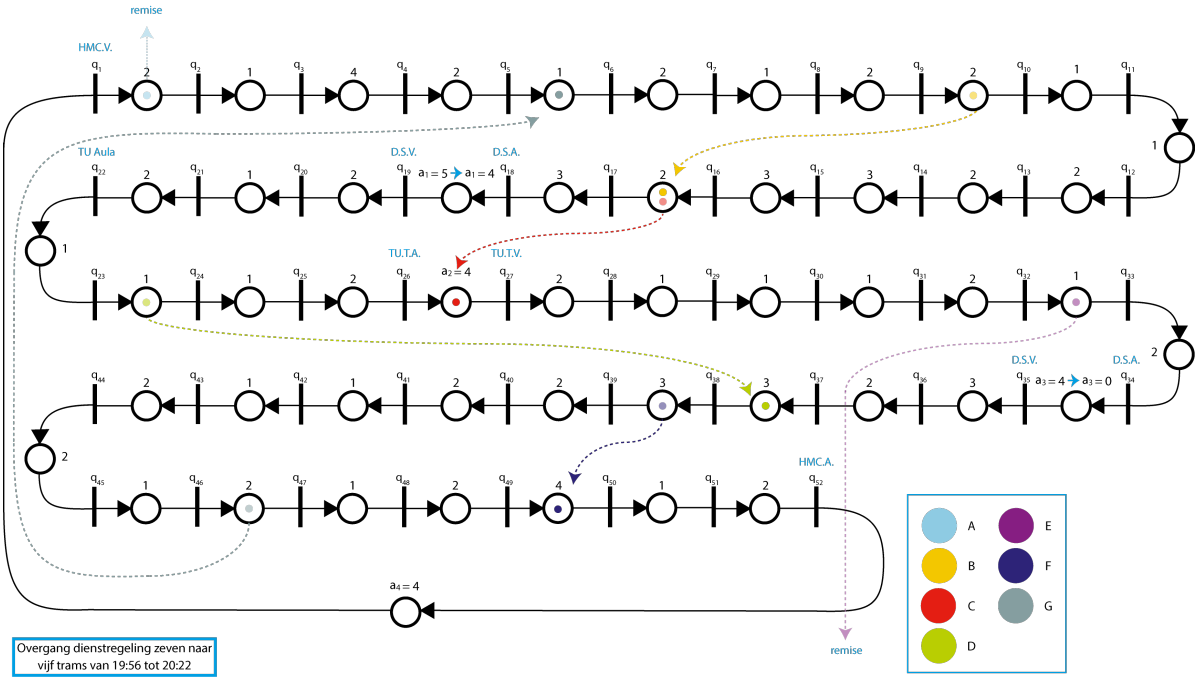
We stellen weer een overgangsmatrix O op, zodanig dat $O \otimes \mathbf{y}(58) = \mathbf{x}(45)$. We krijgen dan een grote diagonaalmatrix O van 52 x 52. Deze valt terug te vinden in appendix C. Hieronder geven we een verkorte weergave met de vectoren $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ en de matrix $\tilde{O}$. De vectoren $\mathbf{x}$ en $\mathbf{y}$ zijn voor deze stap verkort tot de vectoren :

$$\tilde{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_5 \\ x_8 \\ x_{15} \\ x_{22} \\ x_{25} \\ x_{29} \\ x_{36} \\ x_{37} \\ x_{43} \\ x_{47} \end{pmatrix}, \tilde{\mathbf{x}}(45) = \begin{pmatrix} 20:13 \\ 20:22 \\ 20:06 \\ 20:19 \\ 20:16 \\ 20:19 \\ 20:08 \\ 20:18 \\ 20:20 \\ 20:12 \\ 20:19 \end{pmatrix} \quad \tilde{\mathbf{y}} = \begin{pmatrix} y_1 \\ y_5 \\ y_8 \\ y_{15} \\ y_{22} \\ y_{25} \\ y_{29} \\ y_{36} \\ y_{37} \\ y_{43} \\ y_{47} \end{pmatrix}, \tilde{\mathbf{y}}(58) = \begin{pmatrix} 19:56 \\ 19:50 \\ 19:54 \\ 19:52 \\ 19:55 \\ 19:43 \\ 19:52 \\ 19:51 \\ 19:38 \\ 19:50 \\ 19:42 \end{pmatrix} \tag{3.18}$$

De reden dat we dit doen is omdat $\mathbf{x}'$ en $\mathbf{y}'$ niet dezelfde grootte hadden en we dus niet een $\otimes$-operatie met een overgangsmatrix $\tilde{O}$ konden gebruiken om $\tilde{O} \otimes \mathbf{y}'(58) = \mathbf{x}'(45)$ weer te geven. De oorspronkelijke vectoren $\mathbf{x}$ en $\mathbf{y}$ hebben immers ook dezelfde grootte.
We laten hieronder zien hoe zo'n overgangsmatrix $\tilde{O}$ eruit kan zien voor $\tilde{O} \otimes \tilde{\mathbf{y}}(58) = \tilde{\mathbf{x}}(45)$:

$$\tilde{O} = \begin{pmatrix} 17 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\ \epsilon & 32 & \ddots & & & & & & & & \vdots \\ \vdots & \ddots & 12 & \ddots & & & & & & & \vdots \\ \vdots & & \ddots & 27 & \ddots & & & & & & \vdots \\ \vdots & & & \ddots & 21 & \ddots & & & & & \vdots \\ \vdots & & & & \ddots & 36 & \ddots & & & & \vdots \\ \vdots & & & & & \ddots & 16 & \ddots & & & \vdots \\ \vdots & & & & & & \ddots & 27 & \ddots & & \vdots \\ \vdots & & & & & & & \ddots & 42 & \ddots & \vdots \\ \vdots & & & & & & & & \ddots & 22 & \epsilon \\ \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & \cdots & 37 \end{pmatrix} \tag{3.19}$$

Alle punten in de matrix staan weer voor het element ϵ.
Met de originele overgangsmatrix krijgen we $O \otimes \mathbf{y}(58) = \mathbf{x}(45)$ rond 20 uur. Hierna zal de $A \otimes \mathbf{x}(k)$ weer worden toegepast om de vertrektijden van $\mathbf{x}(k+1)$ te berekenen voor de dienstregeling voor vijf trams
Om duidelijk te maken wat er precies in deze stappen zijn gebeurd, kunt u hieronder een situatieweergave van het Petrinetwerk van het traject van tramlijn 19 beschouwen. Hierin zult u zien hoe de dienstregeling met zeven trams op 19:56 over zal gaan naar een dienstregeling van vijf trams op 20:22. Let op: de waarden van a_1 en a_3 veranderen hier ook! Dit komt omdat we een andere eis stelden aan de trajectduur bij een dienstregeling met vijf trams dan met zeven.
Een grotere afbeeldingen hiervan kunt u in de appendix vinden.



Figuur 3.4: Overzichtsweergave van 19:56 tot 20:22 waarin er wordt overgegaan op een nieuwe dienstregeling

Hierin valt het volgende te zien:

- Op 19:56 zijn er nog zeven trams in het traject aanwezig. Deze staan op de plaatsen (of trajecten) tussen de haltes $q_1 - q_2$; $q_9 - q_{10}$; $q_{16} - q_{17}$; $q_{23} - q_{24}$; $q_{32} - q_{33}$; $q_{38} - q_{39}$ en $q_{46} - q_{47}$
- Vervolgens doen de trams tussen 19:56 en 20:22 het volgende:
  - $A \rightarrow$ remise: tram A staat op 19:56 klaar om vanaf HMC te vertrekken. In plaats daarvan vertrekt hij naar de remise en verdwijnt uit het traject.
  - $B \rightarrow q_{16}$: tram B vertrekt op 19:54 vanaf de halte q_8. Twee minuten later, op 19:56 vertrekt hij dan vanaf q_9. 16 minuten later, om 20:12 komt B aan op de halte q_{16}. Hier wacht B tien minuten voordat hij verder rijdt via de daluren dienstregeling.
  - $C \rightarrow q_{26}$: op 19:56 rijdt tram C al op het traject tussen q_{16} en q_{17}. Na 18 minuten, op 20:14 komt C aan op de halte q_{26}. Naast de reguliere wachttijd van vier minuten die hier geldt, wacht C een extra drie minuten om vervolgens om 20:21 richting q_{27} te vertrekken.
  - $D \rightarrow q_{37}$: op 19:56 vertrekt tram D vanaf q_{23}. 25 minuten later, om 20:21 komt D aan op q_{37} en wacht hier vervolgens één minuut om op 20:22 te vertrekken richting q_{38}.
  - $E \rightarrow$ remise: op 19:56 rijdt tram E weg vanuit q_{32}. Na drie minuten, om 19:59 komt hij aan op de halte q_{34} (Delft station). Hier worden alle reizigers verzocht uit te stappen en vertrekt de tram verder richting q_{52} (HMC), om vanuit daar naar de remise te rijden. Onderweg is het niet mogelijk voor passagiers om nog in te stappen. 35 minuten later komt E aan op q_{52} (HMC) en rijdt om 20:34 richting de remise.
  - $F \rightarrow q_{50}$: op 19:56 vertrekt tram F vanaf q_{38} naar q_{49}. Na 19 minuten, om 20:15 komt F aan op q_{49} en wacht vervolgens zeven minuten om om 20:22 hier weer weg te rijden volgens de daluren dienstregeling.
  - $G \rightarrow q_5$: G vertrekt om 19:56 vanaf q_{46}. Na 24 minuten komt deze aan op de halte q_5 om 20:20. Na twee minuten wachten vertrekt G hier om 20:22 richting q_6
- Om 20:22 bevinden de trams zich zoals aangegeven aan het einde van de peilen in figuur 3.4.Tram A en E bevinden zich vanaf 20:38 niet meer in het traject.

3.3.4. Conclusie hoofdstuk 3

In dit hoofdstuk hebben we uitgezocht hoe we het beste in een dienstregeling kunnen overstappen van spits- naar daluren. We hebben dit eerst aan de hand van een klein traject gedaan. Hierna zijn we gaan kijken naar het toekomstige traject van tramlijn 19. Daarbij hebben we analoog aan hoofdstuk 2 een dienstregeling voor zeven trams opgesteld. Hierbij is rekening gehouden met een aantal voorwaarden. Vervolgens hebben we een overgangsmatrix bepaald voor de overgang van de (spits)dienstregeling van zeven trams naar de (dal)dienstregeling van vijf trams.
In hoofdstuk 4 zullen we een volledige dienstregeling voor tramlijn 19 presenteren.

4

Samenvatting en resultaten

In dit project zochten we naar een manier om de dienstregeling van tramlijn 19 uit te breiden. In 2020 zullen de haltes op de TU Campus ook worden aangedaan door deze tramlijn en is er dus een nieuwe dienstregeling nodig.
Deze dienstregeling met haar voorwaarden is onderzocht in de vorige hoofdstukken. Wat hierin is gevonden kunnen we samenvatten tot het volgende:

- Er is een dienstregeling voor zeven trams opgesteld. Deze trams doen elke 15 minuten een halte aan. Tramlijn 19 rijdt vanaf 5:26 's ochtends tot 20 uur 's avonds via deze dienstregeling. We noemen dit ook wel de spitsuren. Na 20 uur wordt er overgegaan op een dienstregeling met vijf trams. De vertrektijden hiervan werden berekend aan de hand van $\mathbf{y}(k+1) = B \otimes \mathbf{y}(k)$ (zie hoofdstuk 3.3.1).

- Er is een dienstregeling voor vijf trams opgesteld. Deze trams doen elke 20 minuten een halte aan en is bedoeld als dienstregeling voor 's avonds na 20 uur. We noemen dit ook wel de daluren. De vertrektijden hiervan werden berekend aan de hand van $\mathbf{x}(k+1) = A \otimes \mathbf{x}(k)$ (zie hoofdstuk 2.3). Na 00:49 rijden er geen trams meer over het traject.

- De halte HMC Anthoniushoeve dient als remise. In het huidige traject verlaten ook de trams het traject via deze halte.

- Er is rekening gehouden met een aantal voorwaarden in het model:

  - De eerste tram moet rond 5:30 vanaf HMC Anthoniushoeve richting TU Technopolis vertrekken
  - Rond 8:25 en 8:40 moeten er trams de halte TU Aula aandoen, om studenten op tijd bij hun college te kunnen laten aankomen
  - Op de haltes TU Technopolis, HMC Anthoniushoeve en Delft station (in de richting HMC- TU Technopolis) dient een minimale wachttijd van vier minuten te zijn.
  - Rond 01:00 's nachts moet de laatste tram aankomen bij HMC Anthoniushoeve om vervolgens in de remise gestald te worden
  - De trajecttijden (of plaatsen) staan vast. Trams kunnen dus niet versnellen.

Dit heeft geresulteerd in een dienstregeling voor tramlijn 19 uitgebreid met haltes op de TU Campus. De vertrektijden van een aantal haltes worden hier gegeven. Elke andere vertrektijd kan berekend worden met behulp van Python (zie appendix).
Hieronder ziet u bijvoorbeeld de vetrektijden van de halte HMC Anthoniushoeve (vertrek) en TU Aula, ofwel respectievelijk q_1, q_{22} en q_{31} in ons model. Zo kunnen we van elke halte de vertrektijden op een dag opstellen.

Tabel 4.1: Vertrektijden halte HMC Anthoniushoeve

richting TU Technopolis																				
5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**0**
26	11	11	11	11	11	11	11	11	11	11	11	11	11	11	13	13	13	13		
41	26	26	26	26	26	26	26	26	26	26	26	26	26	26	33	33	33			
56	41	41	41	41	41	41	41	41	41	41	41	41	41	41	53	53	53			
	56	56	56	56	56	56	56	56	56	56	56	56	56	56						

Tabel 4.2: Vetrektijden van TU Aula

richting TU Technopolis																				
5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**0**
	10	10	10	10	10	10	10	10	10	10	10	10	10	10	16	16	16	16		
	25	25	25	25	25	25	25	25	25	25	25	25	25	25	36	36	36	36		
	40	40	40	40	40	40	40	40	40	40	40	40	40	40	56	56	56	56		
	55	55	55	55	55	55	55	55	55	55	55	55	55	55						
richting HMC Anthoniushoeve																				
5	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**0**
	24	9	9	9	9	9	9	9	9	9	9	9	9	9	10	10	10	10	10	
	39	24	24	24	24	24	24	24	24	24	24	24	24	24	30	30	30	30		
	54	39	39	39	39	39	39	39	39	39	39	39	39	39	50	50	50	50		
		54	54	54	54	54	54	54	54	54	54	54	54	54						

5

Discussie

In dit verslag is er een dienstregeling voor tramlijn 19 bepaald. Er is aan de voorwaarden die gesteld zijn voldaan. Ook is er een overgang tussen een daluren- en een spitsuren dienstregeling gevonden. Al met al is wat we wilden onderzoeken ook daadwerkelijk berekend.
Toch zijn er wat verbeterpunten en aanvullingen. De tijd die je in dit project hebt is vrij kort en er zijn een aantal zaken die je met meer tijd of eventueel een vervolg project verder zou kunnen onderzoeken. Een goed voorbeeld hiervan is de overgang tussen zeven en vijf trams. Op dit moment duurt deze overgang van 19:56 tot 20:22, een vrij lang tijdsbestek. Hierdoor moeten sommige trams een tijdje wachten op een halte. Het is misschien mogelijk dit in een kortere tijdsspan te doen. Ook is er in dit project alleen rekening gehouden met de overgang van spitsuren naar daluren, maar in dienstregelingen van andere vervoersmiddelen zouden we dit misschien ook omgekeerd willen doen. Bij tramlijn 19 zien we echter in de huidige dienstregeling dat er wordt begonnen met een spitsuren dienstregeling en geëindigd wordt met een daluren dienstregeling. Deze voorwaarde is in dit project overgenomen voor de toekomstige dienstregeling. Er was daarom in dit onderzoek geen aanleiding meer een overgang van daluren naar spitsuren te onderzoeken. Dit zou misschien wel voor een vervolgonderzoek interessant kunnen zijn!

Bibliografie

[1] Heidergott, B., Olsder, G.J., & van der Woude, J.W. (2006). *Max Plus at work: Modeling and analysis of synchronized systems: A course on Max-Plus algebra and its applications.* Princeton: Princeton University Press.

[2] Subiono (2000). *On classes of min-max-plus systems and their applications.* Delft: Delft University Press.

[3] Schaft, A.V. van der (2010). Characterization and partial synthesis of the behavior of resistive circuits at their terminals. *Systems & Control Letters, 59*(7), 423-428. doi:10.1016/j.sysconle.2010.05.005

[4] Lopes, G.A., Babuška, R., Schutter, B. De., & A.J.J. van den Boom. (2009). Switching Max-Plus models for legged locomotion. *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO).* doi:10.1109/robio.2009.5420626

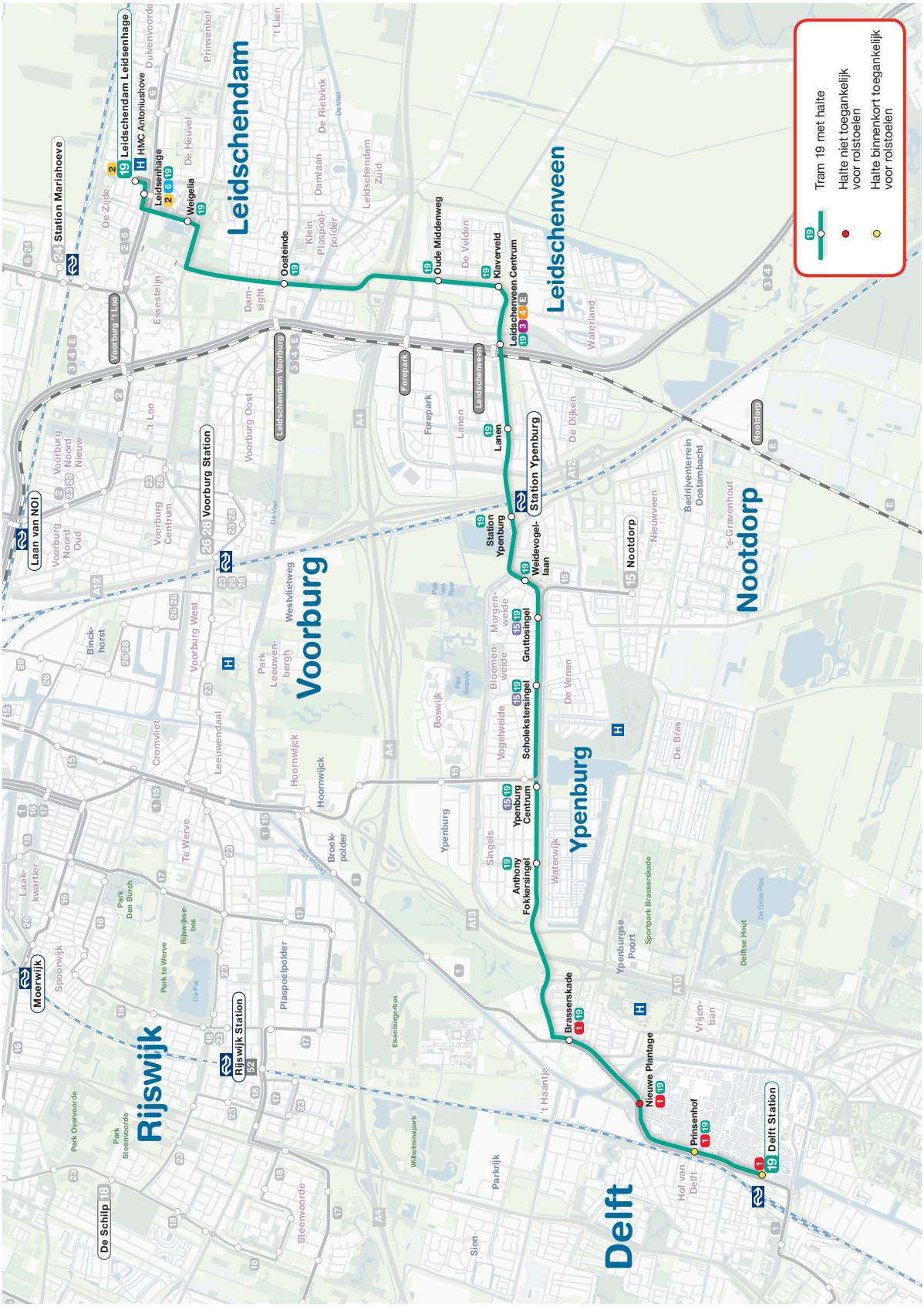
[5] HTM_Tram19-web [Digital image]. (December 9). Retrieved May 30, 2018, from `https://cdn.chainels.com/image/311512833077396817`

[6] Tram 19 - HTM. (n.d.). Retrieved May 30, 2018, from `https://www.htm.nl/reisinformatie/tram-19/`

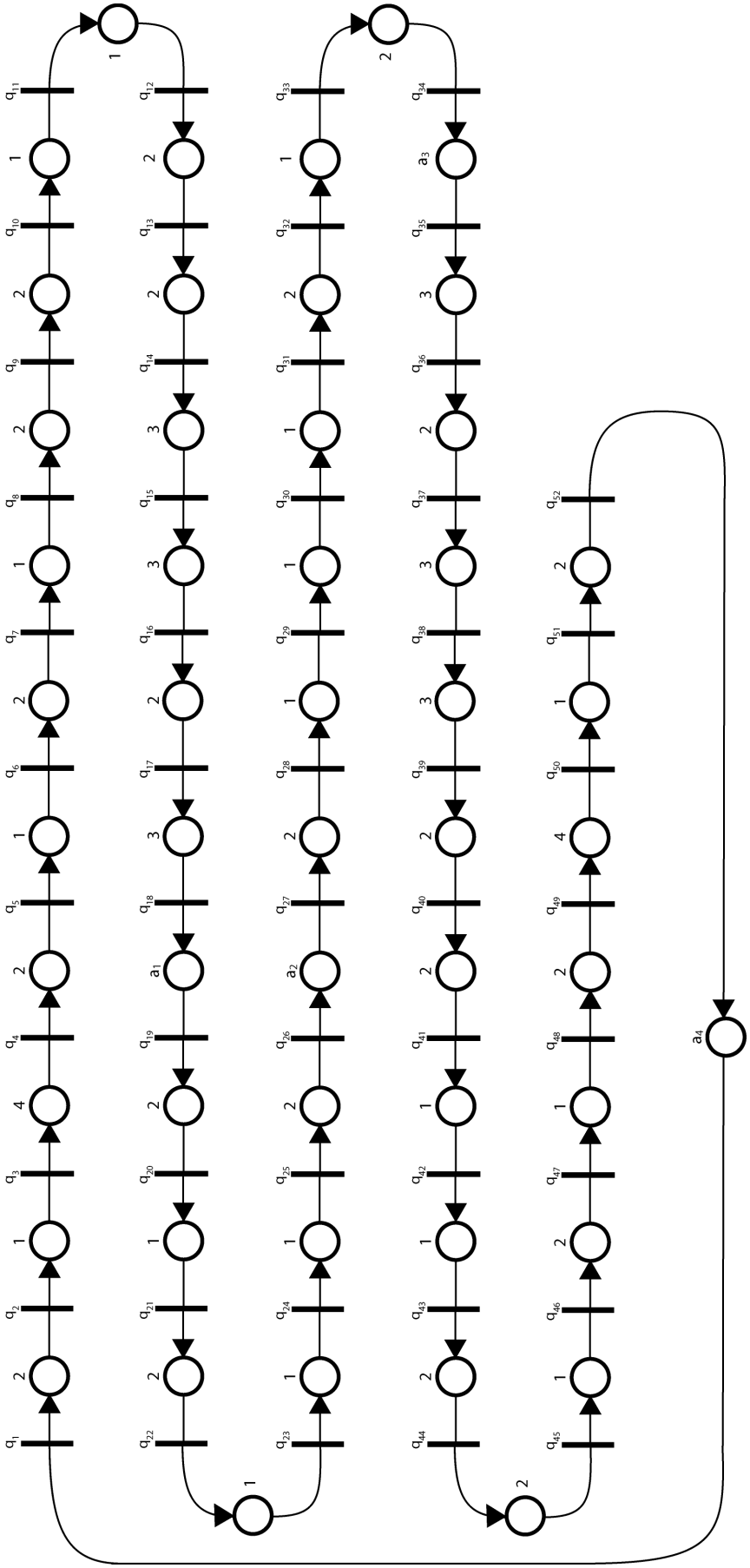
[7] Tramlijn 19 (Haaglanden). (2018, May 10). Retrieved May 30, 2018, from `https://nl.wikipedia.org/wiki/Tramlijn_19_(Haaglanden)`

A

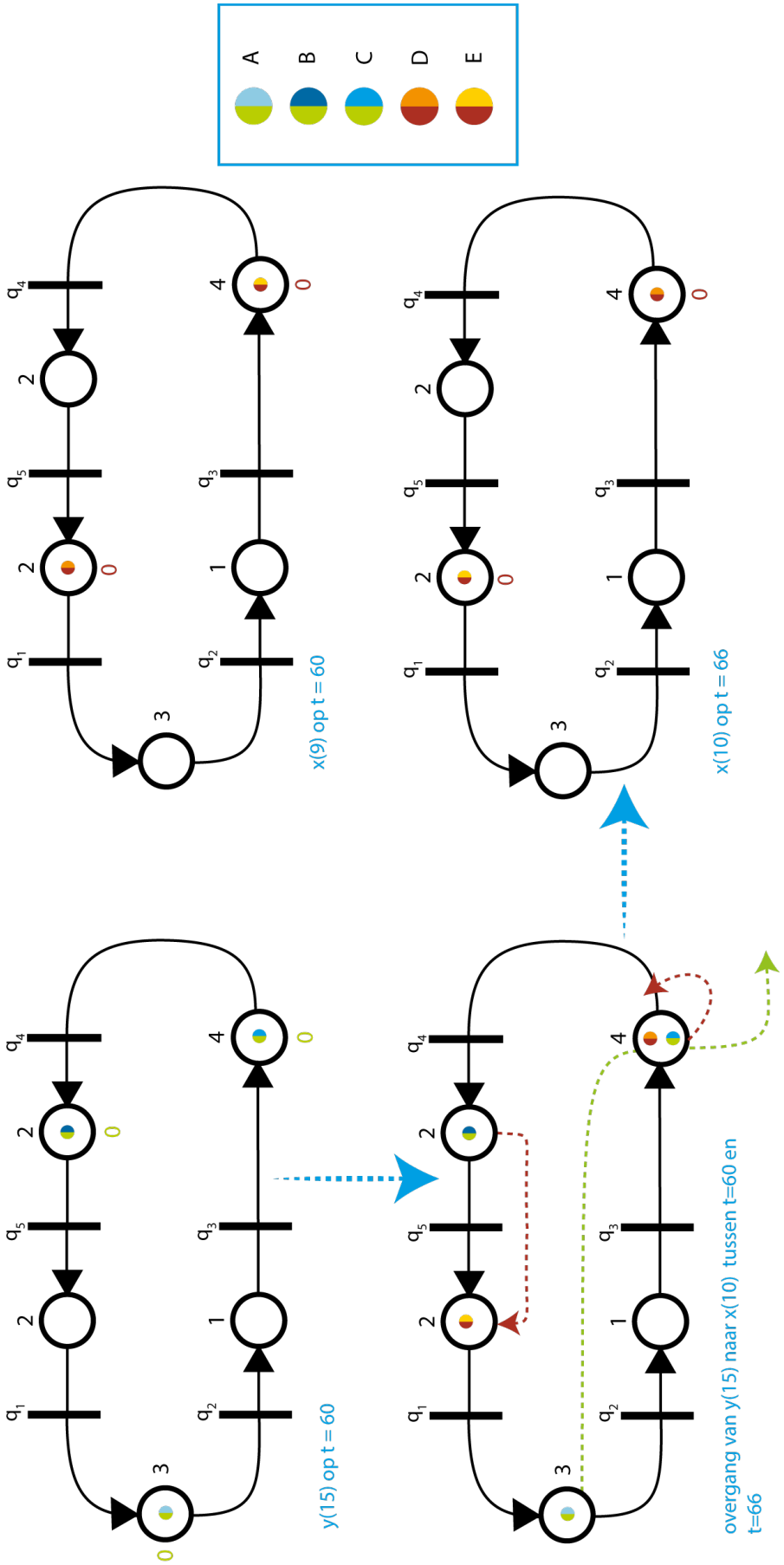
Hieronder volgen verschillende afbeeldingen uit het verslag in uitvergrootte weergave.



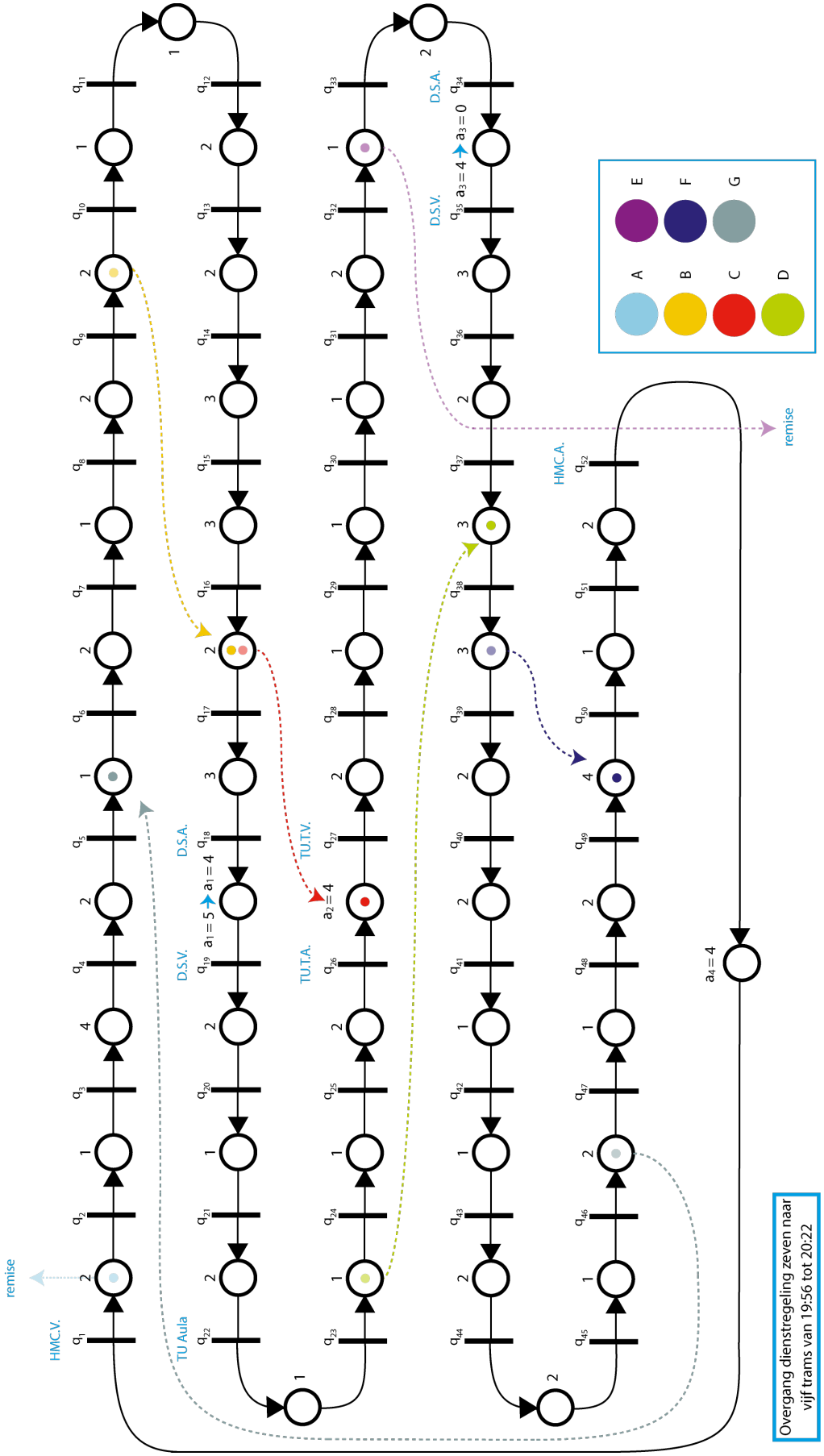
Figuur A.1: Huidig traject tram 19 [5]



Figuur A.2: Petrinetwerk van alle haltes van tramlijn 19



Figuur A.3: Overgang drie naar twee trams in versimpeld traject



Figuur A.4: Overgang dienstregeling zeven trams naar vijf trams tussen 19:56 en 20:22

B

Hieronder volgt de Python code die is gemaakt voor het traject met vijf en zeven trams. Ook de Python code voor de overgang in het versimpelde circuit van lengte twaalf vindt u hier.

B.1. Overgang simpel traject

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu May 24 12:19:17 2018

@author: DoortjeVisser
"""

import numpy as np

################################################################################
def departures2(x, m, n):
    #calculates the departure times in circuit of lenght 12 and 2 trams
    print("the departure times of k = ", n, "are:", "\n", x)
    print("-------------------------------------------------------")
    values = [x]

    while n < 19:
        #calculates the new departure times with the matrix A2
        y_2 = x.item(1)
        y_5 = x.item(4)

        d_1 = y_5 + 2
        d_2 = y_5 + 5
        d_3 = y_2 + 1
        d_4 = y_2 + 5
        d_5 = y_2 + 7

        if d_1 and d_2 and d_3 and d_4 and d_5 > 60:
            m = m + 1
            d_1 = np.mod(d_1, 60)
            d_2 = np.mod(d_2, 60)
            d_3 = np.mod(d_3, 60)
            d_4 = np.mod(d_4, 60)
            d_5 = np.mod(d_5, 60)
        d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5]])
        # delivers an array with the hours and minutes of departures
        d_m = np.array([[str(m) + ":"+ str(d.item(0))], [str(m) + ":"+ str(d.item(1))],\
                        [str(m) + ":"+ str(d.item(2))], [str(m) + ":"+ str(d.item(3))], \
                        [str(m) + ":"+ str(d.item(4))]])

        print("the departure times of k = ", n+1, "are:", "\n", d_m)
        print("-------------------------------------------------------")
        x = d
        values.append(d)
```

```
        n = n + 1
    else:
        print("we have to stop now")
        return values

def departures3(x,m,n):
    #calculates the departure times in circuit of lenght 12 and 2 trams
    #n= 1
    #m = 12
    print("the departure times of k = ",n, "are:", "\n", x)
    print("-------------------------------------------------------")
    values = [x]
    while n < 16 or  n in range(30, 46):
        #calculates the new departure times with the matrix A3
        y_1 = x.item(0)
        y_3 = x.item(2)
        y_4 = x.item(3)

        d_1 = int(y_4 + 4)
        d_2 = int(y_1 + 3)
        d_3 = int(y_1 + 4)
        d_4 = int(y_3 + 4)
        d_5 = int(y_4 + 2)

        if d_1 and d_2 and d_3 and d_4 and d_5 > 60:
            m = m + 1
            d_1 = np.mod(d_1, 60)
            d_2 = np.mod(d_2, 60)
            d_3 = np.mod(d_3, 60)
            d_4 = np.mod(d_4, 60)
            d_5 = np.mod(d_5, 60)
        d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5]])
        # delivers an array with the hours and minutes of departures
        d_m = np.array([[str(m) + ":"+ str(d.item(0))], [str(m) + ":"+ str(d.item(1))],\
                        [str(m) + ":"+ str(d.item(2))], [str(m) + ":"+ str(d.item(3))], \
                        [str(m) + ":"+ str(d.item(4))]])

        print("the departure times of k = ", n+1, "are:", "\n", d_m)
        print("-------------------------------------------------------")

        x = d
        values.append(d)
        n = n +1
    else:
        print("we have to stop now")
        return values


def converter_3_to_2(x):
    d_1 = x.item(0) + 2
    d_2 = x.item(1) + 6
    d_3 = x.item(2) + 0
    d_4 = x.item(3) + 4
    d_5 = x.item(4) + 8
    d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5]])
    return np.mod(d, 60)

def converter_2_to_3(x):
    d_1 = x.item(0) + 8
    d_2 = x.item(1) + 4
    d_3 = x.item(2) + 10
    d_4 = x.item(3) + 6
    d_5 = x.item(4) + 2
    d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5]])
    return np.mod(d, 60)



################################################################################

# MAIN PROGRAMM
```

```
117
p = np.zeros((5,1))
119 q = np.array([[2], [5], [0], [4], [6]])

121 #print("the departure times in a circuit of 12 with 2 trams is:")
#q_values = departures2(q, 1, 12)
123 #print("\n")
#print("the departure times in a circuit of 12 with 3 trams is:")
125
print("from 12 to 13 h. the departure times are:", "n")
127 p_values1 = departures3(p,12,1)
w = converter_3_to_2(p_values1[15])
129 print("from 13 to 14 h. the departure times are:", "\n")
q_values = departures2(w, 13, 10)
131 print("from 14 to 15 h. the departure times are:", "\n")
z = converter_2_to_3(q_values[9])
133 p_values2 = departures3(z, 14, 31)

135 #testvalues2 = departures2(q, 12, 1)
#testvalues3 = departures3(p, 12, 1)
```

python/circuit_12.py

B.2. Vijf trams

B.2.1. Vertrektijden

```
1 #!/usr/bin/env python3
# -*- coding: utf-8 -*-
3 """
Created on Mon May 14 16:13:30 2018
5
@author: DoortjeVisser
7 """

9 import numpy as np

11 def departures(x,m):
    #calculates next departure times of q1-q52
13     n = 1
    counter = 1
15     values = []
    while n < 61:
17         print ("the departure times for k =", str(n), "are:")
        print("-----------------------------------")
19         #calculates the new departure time x1-x52
        y_5 = x.item(4)
21         y_15 = x.item(14)
        y_25 = x.item(24)
23         y_37 = x.item(36)
        y_47 = x.item(46)
25
        d_1 = y_47 + 14
27         d_2 = y_47 + 16
        d_3 = y_47 + 17
29         d_4 = y_47 + 21
        d_5 = y_47 + 23
31
        d_6 = y_5 + 1
33         d_7 = y_5 + 3
        d_8 = y_5 + 4
35         d_9 = y_5 + 6
        d_10 = y_5 + 8
37         d_11 = y_5 + 9
        d_12 = y_5 + 10
39         d_13 = y_5 + 12
        d_14 = y_5 + 14
41         d_15 = y_5 + 17

43         d_16 = y_15 + 3
```

```
        d_17 = y_15 + 5
        d_18 = y_15 + 8
        d_19 = y_15 + 12
        d_20 = y_15 + 14
        d_21 = y_15 + 15
        d_22 = y_15 + 17
        d_23 = y_15 + 18
        d_24 = y_15 + 19
        d_25 = y_15 + 20

        d_26 = y_25 + 2
        d_27 = y_25 + 6
        d_28 = y_25 + 8
        d_29 = y_25 + 9
        d_30 = y_25 + 10
        d_31 = y_25 + 11
        d_32 = y_25 + 13
        d_33 = y_25 + 14
        d_34 = y_25 + 16
        d_35 = y_25 + 16
        d_36 = y_25 + 19
        d_37 = y_25 + 21

        d_38 = y_37 + 3
        d_39 = y_37 + 6
        d_40 = y_37 + 8
        d_41 = y_37 + 10
        d_42 = y_37 + 11
        d_43 = y_37 + 12
        d_44 = y_37 + 14
        d_45 = y_37 + 16
        d_46 = y_37 + 17
        d_47 = y_37 + 19

        d_48 = y_47 + 1
        d_49 = y_47 + 3
        d_50 = y_47 + 7
        d_51 = y_47 + 8
        d_52 = y_47 + 10

        d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5], [d_6], [d_7], \
                      [d_8], [d_9], [d_10], [d_11], [d_12], [d_13], \
                      [d_14], [d_15], [d_16], [d_17], [d_18], [d_19], \
                      [d_20], [d_21], [d_22], [d_23], [d_24], [d_25], \
                      [d_26], [d_27], [d_28], [d_29], [d_30], [d_31], \
                      [d_32], [d_33], [d_34], [d_35], [d_36], [d_37], \
                      [d_38], [d_39], [d_40], [d_41], [d_42], [d_43], \
                      [d_44], [d_45], [d_46], [d_47], [d_48], [d_49], \
                      [d_50], [d_51], [d_52]])

        if np.median(d) >= counter * 60:
            m = m + 1
            counter = counter + 1


        #a_d = list(d-x)
        #a_d_r = list(np.around(np.array(a_d),0))
        #print("d=", d)
#       print("new departure time for k= ", str(n), "is:", "\n")
#       print(d, "\n")
# ================================================================================
#         print("most important departure times are", "\n", \
#               "x1 = ", d_1, "\n", \
#               "x18 =", d_18, "\n", \
#               "x19 =", d_19, "\n", \
#               "x22 =", d_22, "\n", \
#               "x26 =", d_26, "\n", \
#               "x27 =", d_27, "\n", \
#               "x34 =", d_34, "\n", \
#               "x35 =", d_35, "\n", \
#               "x52 =", d_52, "\n")
```

```
# ============================================================================
        print("most important departure times are:", "\n", \
         "x1 = ", str(m) + ":",np.mod(d_1,60), "\n", \
         "x8 = ", str(m) + ":",np.mod(d_18,60), "\n", \
         "x15 =", str(m) + ":",np.mod(d_15,60), "\n", \
         "x18 =", str(m) + ":",np.mod(d_18,60), "\n", \
         "x19 =", str(m) + ":",np.mod(d_19,60), "\n", \
         "x22 =", str(m) + ":",np.mod(d_22,60), "\n", \
         "x26 =", str(m) + ":",np.mod(d_26,60), "\n", \
         "x27 =", str(m) + ":",np.mod(d_27,60), "\n", \
         "x29 =", str(m) + ":",np.mod(d_29,60), "\n", \
         "x34 =", str(m) + ":",np.mod(d_34,60), "\n", \
         "x35 =", str(m) + ":",np.mod(d_35,60), "\n", \
         "x36 =", str(m) + ":",np.mod(d_36,60), "\n", \
         "x43 =", str(m) + ":",np.mod(d_43,60), "\n", \
         "x46 =", str(m) + ":",np.mod(d_46,60), "\n", \
         "x52 =", str(m) + ":",np.mod(d_52,60), "\n",)

        values.append(d)
        x = d
        n = n+1

    else:
        return values
        print("we have to stop now")
```

python/alle_haltes_5trams.py

B.2.2. Power algoritme

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  1 21:36:24 2018

@author: DoortjeVisser
"""
import numpy as np
import alle_haltes_5trams as h5
import pandas as pd

##############################################
def PowAlg(lamb,c1, c2, c3, c4, c5):
    lamb_x1 = np.asarray(lamb*4 + c1)
    lamb_x2 = np.asarray(lamb*3 + c2)
    lamb_x3 = np.asarray(lamb*2 + c3)
    lamb_x4 = np.asarray(lamb + c4)
    lamb_x5 = np.asarray(c5)
    new_max_1 = MaxPlusVec(lamb_x1, lamb_x2)
    new_max_2 = MaxPlusVec(new_max_1, lamb_x3)
    new_max_3 = MaxPlusVec(new_max_2, lamb_x4)
    new_max_4 = MaxPlusVec(new_max_3, lamb_x5)
    return new_max_4


def MaxPlusVec(c1,c2):
    #calulates the maximum of the elements of two vectors
    max_c = np.array([])
    for i in range(0, len(c1)):
        max_element = int(max(c1.item(i), c2.item(i)))
        max_c = np.append(max_c, max_element)
    return max_c
#####################################################

x = np.zeros((52,1))
values = h5.departures(x,5)
x1 = values[0]
x2 = values[1]
x3 = values[2]
x4 = values[3]
```

```
x5 = values[4]
v = PowAlg(20, x1, x2, x3, x4, x5)

print("with the Power Algorithm we find the begin vector v =", "\n", v)
v_min = np.asarray(v -min(v))
#v_1 = np.asarray(v_min + 1)
#v_12 = np.asarray(v_min - 12)

print("\n", "we can write v also as: v=", "\n", v_min)

values_v =h5.departures(v_min,5)

#MAKES AN EXCEL LIST OF THE DEPARTURE TIMES
datas = []
for i in range(0,len(values_v)):
    datas.append(list(np.mod(values_v[i],60)))
my_df = pd.DataFrame(data=datas, columns=np.arange(1,53)).T
my_df.to_excel('departures5.xlsx', index=False, header=np.arange(1,len(values_v)+1))
print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][0],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_1_5_trams.xlsx', index=False, header=False)
print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][21],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_22_5_trams.xlsx', index=False, header=False)
print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][30],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_31_5_trams.xlsx', index=False, header=False)
print(my_df)
```

python/power_algoritme_5trams.py

B.3. Zeven trams

B.3.1. Vertrektijden

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri May 18 11:28:51 2018

@author: DoortjeVisser
"""
import numpy as np

def departures7(x,m):
    #calculates next departure times of q1-q52 with 7 trams
    n = 1
    counter = 1
    values = []
    while n < 60:
        print ("the departure times for k =", str(n), "are:")
        print("-----------------------------------------")
        #calculates the new departure time x1-x52
        y_1 = x.item(0)
        y_8 = x.item(7)
```

```
        y_15 = x.item(14)
        y_22 = x.item(21)
        y_29 = x.item(28)
        y_36 = x.item(35)
        y_43 = x.item(42)

        d_1 = y_43 + 21
        d_2 = y_1 + 2
        d_3 = y_1 + 3
        d_4 = y_1 + 7
        d_5 = y_1 + 9
        d_6 = y_1 + 10
        d_7 = y_1 + 12
        d_8 = y_1 + 13

        d_9 = y_8 + 2
        d_10 = y_8 + 4
        d_11 = y_8 + 5
        d_12 = y_8 + 6
        d_13 = y_8 + 8
        d_14 = y_8 + 10
        d_15 = y_8 + 13

        d_16 = y_15 + 3
        d_17 = y_15 + 5
        d_18 = y_15 + 8
        d_19 = y_15 + 13
        d_20 = y_15 + 15
        d_21 = y_15 + 16
        d_22 = y_15 + 18

        d_23 = y_22 + 1
        d_24 = y_22 + 2
        d_25 = y_22 + 3
        d_26 = y_22 + 5
        d_27 = y_22 + 9
        d_28 = y_22 + 11
        d_29 = y_22 + 12

        d_30 = y_29 + 1
        d_31 = y_29 + 2
        d_32 = y_29 + 4
        d_33 = y_29 + 5
        d_34 = y_29 + 7
        d_35 = y_29 + 11
        d_36 = y_29 + 14

        d_37 = y_36 + 2
        d_38 = y_36 + 5
        d_39 = y_36 + 8
        d_40 = y_36 + 10
        d_41 = y_36 + 12
        d_42 = y_36 + 13
        d_43 = y_36 + 14

        d_44 = y_43 + 2
        d_45 = y_43 + 4
        d_46 = y_43 + 5
        d_47 = y_43 + 7
        d_48 = y_43 + 8
        d_49 = y_43 + 10
        d_50 = y_43 + 14
        d_51 = y_43 + 15
        d_52 = y_43 + 17


        d = np.array([[d_1], [d_2], [d_3], [d_4], [d_5], [d_6], [d_7], \
                      [d_8], [d_9], [d_10], [d_11], [d_12], [d_13], \
                      [d_14], [d_15], [d_16], [d_17], [d_18], [d_19], \
                      [d_20], [d_21], [d_22], [d_23], [d_24], [d_25], \
                      [d_26], [d_27], [d_28], [d_29], [d_30], [d_31], \
```

```
                    [d_32], [d_33], [d_34], [d_35], [d_36], [d_37], \
                    [d_38], [d_39], [d_40], [d_41], [d_42], [d_43], \
                    [d_44], [d_45], [d_46], [d_47], [d_48], [d_49], \
                    [d_50], [d_51], [d_52]])

        if np.median(d) >= counter * 60:
            m = m + 1
            counter = counter + 1

        print("most important departure times are:", "\n", \
         "x1 = ", str(m) + ":",np.mod(d_1,60), "\n", \
         "x8 = ", str(m) + ":",np.mod(d_18,60), "\n", \
         "x15 =", str(m) + ":",np.mod(d_15,60), "\n", \
         "x18 =", str(m) + ":",np.mod(d_18,60), "\n", \
         "x19 =", str(m) + ":",np.mod(d_19,60), "\n", \
         "x22 =", str(m) + ":",np.mod(d_22,60), "\n", \
         "x26 =", str(m) + ":",np.mod(d_26,60), "\n", \
         "x27 =", str(m) + ":",np.mod(d_27,60), "\n", \
         "x29 =", str(m) + ":",np.mod(d_29,60), "\n", \
         "x31 =",str(m) + ":", np.mod(d_31,60), "\n", \
         "x34 =", str(m) + ":",np.mod(d_34,60), "\n", \
         "x35 =", str(m) + ":",np.mod(d_35,60), "\n", \
         "x36 =", str(m) + ":",np.mod(d_36,60), "\n", \
         "x43 =", str(m) + ":",np.mod(d_43,60), "\n", \
         "x46 =", str(m) + ":",np.mod(d_46,60), "\n", \
         "x52 =", str(m) + ":",np.mod(d_52,60), "\n", \
         "min(d) =", min(d))
#         print("most important departure times are:", "\n", \
#         "x1 = ", str(m) + ":",d_1, "\n", \
#         "x8 = ", str(m) + ":",d_18, "\n", \
#         "x15 =", str(m) + ":",d_15, "\n", \
#         "x18 =", str(m) + ":",d_18, "\n", \
#         "x19 =", str(m) + ":",d_19, "\n", \
#         "x22 =", str(m) + ":",d_22, "\n", \
#         "x26 =", str(m) + ":",d_26, "\n", \
#         "x27 =", str(m) + ":",d_27, "\n", \
#         "x29 =", str(m) + ":",d_29, "\n", \
#         "x34 =", str(m) + ":",d_34, "\n", \
#         "x35 =", str(m) + ":",d_35, "\n", \
#         "x36 =", str(m) + ":",d_36, "\n", \
#         "x43 =", str(m) + ":",d_43, "\n", \
#         "x52 =", str(m) + ":",d_52, "\n", "minutes", "\n" \
#         "minimum is:", min(d))

        values.append(d)
        x = d
        n = n+1
    else:
        return values
        print("we have to stop now")
```

python/trams_7.py

B.3.2. Power algoritme

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue May 15 16:34:53 2018

@author: DoortjeVisser
"""
import numpy as np
#import importlib
import trams_7 as tr7
#importlib.import_module(.py, alle_haltes)
import pandas as pd


############################################################
```

```
def PowAlg(lamb,c1, c2, c3, c4, c5, c6, c7):
    lamb_x1 = np.asarray(lamb*6 + c1)
    lamb_x2 = np.asarray(lamb*5 + c2)
    lamb_x3 = np.asarray(lamb*4 + c3)
    lamb_x4 = np.asarray(lamb*3 + c4)
    lamb_x5 = np.asarray(lamb*2 + c5)
    lamb_x6 = np.asarray(lamb*1 + c6)
    lamb_x7 = c7
    new_max_1 = MaxPlusVec(lamb_x1, lamb_x2)
    new_max_2 = MaxPlusVec(new_max_1, lamb_x3)
    new_max_3 = MaxPlusVec(new_max_2, lamb_x4)
    new_max_4 = MaxPlusVec(new_max_3, lamb_x5)
    new_max_5 = MaxPlusVec(new_max_4, lamb_x6)
    new_max_6 = MaxPlusVec(new_max_5, lamb_x7)
    return new_max_6


def MaxPlusVec(c1,c2):
    #calulates the maximum of the elements of two vectors
    max_c = np.array([])
    for i in range(0, len(c1)):
        max_element = int(max(c1.item(i), c2.item(i)))
        max_c = np.append(max_c, max_element)
    return max_c

############################################################

x = np.zeros((52,1))
values = tr7.departures7(x,5)
x1 = values[0]
x2 = values[1]
x3 = values[2]
x4 = values[3]
x5 = values[4]
x6 = values[5]
x7 = values[6]
v = PowAlg(15, x1, x2, x3, x4, x5, x6, x7)

print("with the Power Algorithm we find the begin vector v =", "\n", v)

v_min = np.asarray(v -min(v))
v_7 = np.asarray(v_min + 7)

print("\n", "we can write v also as: v=", "\n", v_7)
print("most important departure times are", "\n", \
#       "v1 = ", "5:", v_min[0], "\n", \
#       "v5 = ", "5:", v_min[4], "\n", \
#       "v15 = ", "5:",v_min[14], "\n", \
#       "v25 = ", "5:", v_min[24], "\n", \
#       "v37 = ", "5:", v_min[36], "\n", \
#       "v47 = ", "5:", v_min[46], "\n")

#       "v1 = ", "5:", np.mod(v_min[0],60), "\n", \
#       "v5 = ", "5:", np.mod(v_min[4],60), "\n", \
#       "v15 = ", "5:",np.mod(v_min[14],60), "\n", \
#       "v25 = ", "5:", np.mod(v_min[24],60), "\n", \
#       "v37 = ", "5:", np.mod(v_min[36],60), "\n", \
#       "v47 = ", "5:", np.mod(v_min[46],60), "\n")

      "v1 = ", "5:", v_7[0], "\n", \
      "v5 = ", "5:", v_7[4], "\n", \
      "v15 = ", "5:",v_7[14], "\n", \
      "v25 = ", "5:", v_7[24], "\n", \
      "v37 = ", "5:", v_7[36], "\n", \
      "v46 =", "5", v_7[45], "\n",\
      "v47 = ", "5:", v_7[46], "\n")

print("\n", "now we calculate the departure times with this begin vector v:")

values_v = tr7.departures7(v_7,5)

```

```
#MAKES AN EXCEL LIST OF THE DEPARTURE TIMES
res = []
for i in range(0,len(values_v)):
    res.append(list(np.mod(values_v[i],60)))
my_df = pd.DataFrame(data=res, columns=np.arange(1,53)).T
my_df.to_excel('departures.xlsx', index=False, header=np.arange(1,len(values_v)+1))
#print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][0],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_1_7_trams.xlsx', index=False, header=False)
#print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][21],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_22_7_trams.xlsx', index=False, header=False)
#print(my_df)

firstdep = []
for i in range(0,len(values_v)):
    firstdep.append(list(np.mod(values_v[i][30],60)))
my_df = pd.DataFrame(data=firstdep)
#, columns=np.arange(1,53)).T
my_df.to_excel('halte_31_7_trams.xlsx', index=False, header=False)
print(my_df)



#res = [[list(v_7)]]
#for k in range(0,len(values_v)):
##    res.append("k="+str(k+1))
#    res.append(list(values_v[k]))
#my_df = pd.DataFrame(res)
#my_df.to_csv('out.csv', index=False, header=False)
#print(my_df)
```

python/power_algorithm_7_trams.py

C

Hier volgt de overgangsmatrix berekend in hoofdstuk 3.3.3

$$
\begin{array}{c}
 \\ \{1-10\} \\ \{11-20\} \\ \{21-30\} \\ \{31-40\} \\ \{41-50\} \\ \{51-52\}
\end{array}
\begin{array}{c}
\begin{array}{cccccc} [1-10] & [11-20] & [21-30] & [31-40] & [41-50] & [51-52] \end{array} \\
\begin{pmatrix}
O_1 & \epsilon & \cdots & \cdots & \cdots & \epsilon \\
\epsilon & O_2 & & & & \vdots \\
\vdots & & O_3 & & & \vdots \\
\vdots & & & O_4 & & \vdots \\
\vdots & & & & O_5 & \vdots \\
\epsilon & \cdots & \cdots & \cdots & \epsilon & O_6
\end{pmatrix}
\end{array}
\tag{C.1}
$$

, met

$$
O_1 = \begin{pmatrix}
17 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\
\epsilon & 32 & \ddots & & & & & & & \vdots \\
\vdots & \ddots & 32 & \ddots & & & & & & \vdots \\
\vdots & & \ddots & 32 & \ddots & & & & & \vdots \\
\vdots & & & \ddots & 32 & \ddots & & & & \vdots \\
\vdots & & & & \ddots & 12 & \ddots & & & \vdots \\
\vdots & & & & & \ddots & 12 & \ddots & & \vdots \\
\vdots & & & & & & \ddots & 27 & \ddots & \vdots \\
\vdots & & & & & & & \ddots & 27 & \epsilon \\
\epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & 39
\end{pmatrix}, O_2 = \begin{pmatrix}
27 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\
\epsilon & 27 & \ddots & & & & & & & \vdots \\
\vdots & \ddots & 27 & \ddots & & & & & & \vdots \\
\vdots & & \ddots & 27 & \ddots & & & & & \vdots \\
\vdots & & & \ddots & 27 & \ddots & & & & \vdots \\
\vdots & & & & \ddots & 22 & \ddots & & & \vdots \\
\vdots & & & & & \ddots & 22 & \ddots & & \vdots \\
\vdots & & & & & & \ddots & 22 & \epsilon & \vdots \\
\vdots & & & & & & & \ddots & 21 & \epsilon \\
\epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & 21
\end{pmatrix}
$$

$$O_3 = \begin{pmatrix} 21 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\ \epsilon & 21 & \ddots & & & & & & & \vdots \\ \vdots & \ddots & 36 & \ddots & & & & & & \vdots \\ \vdots & & \ddots & 36 & \ddots & & & & & \vdots \\ \vdots & & & \ddots & 36 & \ddots & & & & \vdots \\ \vdots & & & & \ddots & 16 & \ddots & & & \vdots \\ \vdots & & & & & \ddots & 16 & \ddots & & \vdots \\ \vdots & & & & & & \ddots & 16 & \ddots & \vdots \\ \vdots & & & & & & & \ddots & 16 & \epsilon \\ \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & 21 \end{pmatrix}, O_4 = \begin{pmatrix} 21 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\ \epsilon & 21 & \ddots & & & & & & & \vdots \\ \vdots & \ddots & 21 & \ddots & & & & & & \vdots \\ \vdots & & \ddots & 21 & \ddots & & & & & \vdots \\ \vdots & & & \ddots & 27 & \ddots & & & & \vdots \\ \vdots & & & & \ddots & 27 & \ddots & & & \vdots \\ \vdots & & & & & \ddots & 42 & \ddots & & \vdots \\ \vdots & & & & & & \ddots & 22 & \ddots & \vdots \\ \vdots & & & & & & & \ddots & 22 & \epsilon \\ \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & 22 \end{pmatrix}$$

$$O_5 = \begin{pmatrix} 22 & \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon \\ \epsilon & 22 & \ddots & & & & & & & \vdots \\ \vdots & \ddots & 22 & \ddots & & & & & & \vdots \\ \vdots & & \ddots & 37 & \ddots & & & & & \vdots \\ \vdots & & & \ddots & 37 & \ddots & & & & \vdots \\ \vdots & & & & \ddots & 37 & \ddots & & & \vdots \\ \vdots & & & & & \ddots & 37 & \ddots & & \vdots \\ \vdots & & & & & & \ddots & 17 & \ddots & \vdots \\ \vdots & & & & & & & \ddots & 17 & \epsilon \\ \epsilon & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \epsilon & 17 \end{pmatrix}, O_6 = \begin{pmatrix} 5 & \epsilon \\ \epsilon & 5 \end{pmatrix}$$