BACHELOROPDRACHT TECHNISCHE WISKUNDE

Langste afstand per trein binnen een dag

Wat is de maximale afstand die in 24 uur per spoor in Nederland is af te leggen?

Lennard van der Putten

Begeleider:
dr. Walter Kern (DMMP)

22 juni 2016

UNIVERSITEIT TWENTE.

Samenvatting

Dit verslag baseert zich op de onderzoeksvraag *"Wat is de maximale afstand die in een beperkte tijdspanne per spoor in Nederland is af te leggen?"*, in het bijzonder binnen 24 uur. Er wordt een graaf opgesteld waarin alleen de overstapstations zijn opgenomen en dus niet alle kleine tussenstations. De lijnen in de graaf stellen de spoorlijnen voor en hebben elk twee componenten, namelijk de afstand in kilometers en de afstand in tijd. De afgelegde afstand, verstreken tijd, afgelegde route en het huidige punt zullen in een toestand worden bevat en deze toestanden zullen door middel van het Breadth-First Search-algoritme worden doorzocht. Binnen dit algoritme worden de toestanden gefilterd op basis van twee parameters, namelijk de maximale snelheid van een trein en een vooraf vastgestelde streefafstand. Toestanden die in de resterende tijd de streefafstand niet meer kunnen halen, worden uit de toestandenverzameling gehaald. Verder worden er enkele aannames gedaan om de grootte en complexiteit van het model te beperken.

Inhoudsopgave

Hoofdstuk 1

Inleiding

Treinreizen is niet alleen een manier om van A naar B te komen, maar voor vele mensen is het ook een hobby. Zoek een aantal van deze hobbyisten bij elkaar, laat ze een wedstrijd tegen elkaar houden en je hebt een leuk evenement te pakken. Laat één van deze hobbyisten een bachelorstudent wiskunde zijn en je hebt een bacheloropdracht te pakken. Dit is kort gezegd de manier geweest hoe deze bacheloropdracht tot stand is gekomen. In de volgende paragraaf zal de aanleiding voor dit onderzoek nader worden toegelicht.

1.1 Aanleiding van deze bacheloropdracht

Op zaterdag 20 juni 2015 werd de wedstrijd "de Kilometerkampioen" georganiseerd. Het doel was om die dag binnen 24 uur zo veel mogelijk kilometers af te leggen over het Nederlandse spoornetwerk. Hierbij mocht elk traject maximaal één keer worden meegeteld in het aantal kilometers, ook als hetzelfde traject meerdere keren werd afgelegd[1]. Je mocht op een willekeurig tijdstip tussen 0:00 uur en 4:00 uur beginnen, vanaf dat moment had je exact 24 uur. Startte je echter pas na 4:00 uur, dan had je slechts de tijd tot 4:00 uur de volgende nacht.

De winnaar van deze wedstrijd had in totaal 1593 kilometer afgelegd. Ikzelf kwam echter niet verder dan 1309 kilometer, wat dus bijna driehonderd kilometer minder is dan de winnaar. Omdat ik dat gat te groot vind, wil ik met deze bacheloropdracht een wiskundig model maken die op basis van de dienstregeling een planning genereert waarmee de meeste afstand wordt afgelegd binnen een zekere tijdspanne. Vervolgens zal ik dit model implementeren in de programmeertaal Python. De onderzoeksvraag zal geformuleerd worden als volgt: *"Wat is de maximale afstand die in 24 uur per spoor in Nederland is af te leggen?"*, of iets algemener: *"Wat is de maximale afstand die in een beperkte tijdspanne per spoor in Nederland is af te leggen?"*. De exacte formulering van het onderzoek volgt in de volgende paragraaf.

1.2 Formulering van het onderzoek

Het doel van het onderzoek wordt in deze paragraaf wat concreter gemaakt. Gegeven zijn een graaf $G = (V, E)$, waarbij de puntenverzameling V de verzameling van knooppuntstations voorstelt en de lijnenverzameling E de verbindende spoorlijnen tussen de stations, de afstanden d tussen de

[1] Twee uitzonderingen: de trajecten Zwolle–Meppel en Roermond–Sittard telden maximaal twee keer mee.

stations (in kilometers) en de dienstregeling van de treinen. Het doel is nu om een planning te laten genereren waarbij binnen een gegeven tijdsbestek zo veel mogelijk afstand binnen de graaf wordt afgelegd. Hierbij geldt dat elke afstand d slechts één keer per traject meetelt.

Hoofdstuk 2

Opbouw van het model

In dit hoofdstuk zal het probleem in wiskundige termen worden geformuleerd en zullen de bouwstenen voor het model worden vormgegeven.

2.1 Wiskundige formulering van het probleem

Gegeven zijn een graaf $G = (V, E)$ (te zien in figuur A.2), een beginpunt $v_0 \in V$ en een tijdslimiet $T \in \mathbb{R}^+$. Voor alle punten $v, w \in V$ geldt dat de lijn $e = \vec{vw} \in E$ de volgende twee eigenschappen heeft: een afstand $d(v, w) \geq 0$ en de reisduur $t_{\vec{vw}}(x) \geq 0$ vanuit punt v naar punt w vanaf tijdstip x. De afstand $d(v, w)$ is statisch. Hoe we $t_{\vec{vw}}$ bepalen, wordt in paragraaf 2.1.1 besproken.

Nu moeten we een route (Engels: *trail*) $\mathcal{R}_n = \{e_1, e_2, e_3, \ldots, e_n\}$ vinden waarvoor de totale afstand gemaximaliseerd wordt. In formulevorm wordt dit:

$$\max \sum_{i=1}^{n} d(e_i)\mathbb{1}(e_i, \mathcal{R}_{i-1}),$$

waarbij de functie $\mathbb{1}$ gedefinieerd is als

$$\mathbb{1}(e_i, \mathcal{R}_{i-1}) = \begin{cases} 0 & \text{als } e_i \in \mathcal{R}_{i-1}; \\ 1 & \text{als } e_i \notin \mathcal{R}_{i-1}. \end{cases} \tag{2.1}$$

Dit betekent dat de afstand $d(e_i)$ slechts één keer meetelt in de route, ook als traject e_i meerdere keren wordt gepasseerd.

2.1.1 Bepaling eerstvolgende trein

Zij $v, w \in V$ twee stations. Voor elke trein j van punt v naar punt w definiëren we:

- $j(v, w)$ het treinnummer;
- $\alpha_v(j)$ de vertrektijd van trein j uit punt v;
- $\beta_w(j)$ de aankomsttijd van trein j in punt w.

Laat vervolgens de dienstregeling van punt v naar punt w gegeven zijn door de matrix

$$D_{\vec{vw}} = \begin{bmatrix} \alpha_v(j_1) & j_1 & \beta_w(j_1) \\ \alpha_v(j_2) & j_2 & \beta_w(j_2) \\ \alpha_v(j_3) & j_3 & \beta_w(j_3) \\ \vdots & \vdots & \vdots \end{bmatrix},$$

waarbij elke rij één trein voorstelt.
De verzameling van treinen van v naar w (oftewel de verzameling van alle elementen in de middelste kolom van $D_{\vec{vw}}$) noemen we $\mathcal{T}_{\vec{vw}}$:

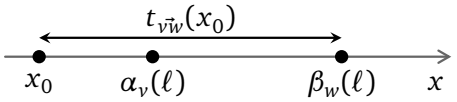
$$\mathcal{T}_{\vec{vw}} := \{j_1,\ j_2,\ j_3,\ \cdots\}.$$

Op een zeker tijdstip x_0 wordt de eerstvolgende trein vanuit v naar station w bepaald door de aankomsttijd in punt w te minimaliseren. Dat wil zeggen: uit de verzameling van treinen die nog moeten vertrekken uit v kiezen we de trein met de vroegste aankomsttijd in w. De eerstvolgende trein van v naar w is dus trein ℓ waarvoor geldt:

$$\ell = \arg\min_{j \in \mathcal{T}_{\vec{vw}}} \{\beta_w(j) \mid \alpha_v(j) \geq x_0\}.$$

Met deze trein ℓ definiëren we de tijdsduur $t_{\vec{vw}}(x_0)$ als de tijd vanaf x_0 tot de aankomsttijd $\beta_w(\ell)$ in punt w, oftewel

$$t_{\vec{vw}}(x_0) = \beta_w(\ell) - x_0.$$



Figuur 2.1: *Visualisatie van de definitie van de variabele $t_{\vec{vw}}(x_0)$.*

2.1.2 Selectie van stations in het model

Nederland telt ruim 400 treinstations. Als we al deze stations verwerken in het model, wordt het model zeer traag en complex, dus zullen we een selectie moeten maken van stations die we in het model gaan gebruiken.
De stations die worden verwerkt (vanaf nu de knooppuntstations genoemd) zijn:

- alle stations van graad 1. Dit zijn de begin- en eindpunten van spoorlijnen. Voorbeelden zijn Glanerbrug, Kampen en Den Helder;
- enkele stations van graad 2. Dit betreft alleen de stations waar treinen beginnen en eindigen en waar redelijkerwijs kan worden overgestapt. Voorbeelden zijn Enschede, Amsterdam Centraal en Alkmaar;
- alle stations van graad 3 of hoger. Dit zijn stations waar tussen diverse trajecten kan worden overgestapt. Voorbeelden zijn Hengelo, Zwolle en Utrecht Centraal.

In totaal houden we 105 stations over, deze zijn te vinden in tabel A.2.
De overige stations zijn kleine tussenstations die niet zullen worden verwerkt in het model. Dit betreft de resterende stations van graad 2 waar niet fatsoenlijk kan worden overgestapt.

2.2 Aannames

Er zullen enkele aannames moeten worden gemaakt om te zorgen dat het model behapbaar blijft. De onderstaande drie aannames worden gemaakt bij het opstellen van het model.

Aanname 2.1. Enkel op de knooppuntstations beschreven in paragraaf 2.1.2 kan worden overgestapt tussen treinen. Een overzicht van alle knooppuntstations vindt u in tabel A.2. In de discussie in hoofdstuk 6 wordt besproken of het verwaarlozen van deze tussenstations grote gevolgen heeft. ◇

Aanname 2.2. Er wordt niet overgestapt op een trein die dezelfde richting op gaat als de trein waar vanuit wordt overgestapt, tenzij deze eerder aankomt op het volgende station. ◇

De derde aanname heeft wat toelichting nodig. Elke trein heeft namelijk een uniek treinnummer. Een trein die een langere route rijdt (bijvoorbeeld $A \to B \to C$) en dus over meerdere trajecten rijdt, heeft op elk van deze trajecten hetzelfde treinnummer. Wanneer je dus eerst van station A naar station B reist en vanaf daar verder reist naar C, en de trein $A \to B$ heeft hetzelfde treinnummer als $B \to C$, dan blijf je in de praktijk in dezelfde trein zitten. Dit brengt ons bij de volgende aanname.

Aanname 2.3. Bij een overstap naar een trein met een ander treinnummer is minimaal twee minuten overstaptijd noodzakelijk. Bij een 'overstap' naar een trein met hetzelfde treinnummer is nul minuten voldoende, aangezien je dan in dezelfde trein blijft zitten. ◇

2.3 Toestanden

We noteren iedere toestand Y_i in de volgende vorm:

$$Y_i = \left(\sum_{k \in \mathcal{R}_i} t_k, \sum_{k \in \mathcal{R}_i} d_k, v_i, \mathcal{R}_i, J_i \right),$$

waarbij t_k de tijdsduur van lijn e_k is (zie paragraaf 2.1.1), d_k de afstand van lijn e_k, v_i het huidige station, $\mathcal{R}_i$ de afgelegde route als verzameling van gepasseerde lijnen en J_i het treinnummer van de laatste trein die in route $\mathcal{R}_i$ genomen is. Deze J_i is nodig om de overstaptijd naar de eerstvolgende trein in de volgende toestand te kunnen bepalen vanwege aanname 2.3.
Ten behoeve van de notatie noemen we de eerste term (de verstreken tijd) x_i en de tweede term (de afgelegde afstand) s_i. Dan krijgen we dus:

$$Y_i = (x_i, s_i, v_i, \mathcal{R}_i, J_i).$$

Daaruit volgt dat de begintoestand Y_0 met een gegeven startpunt v_0 gedefinieerd is als

$$Y_0 = (0, 0, v_0, \emptyset, J_0),$$

waarbij J_0 omwille van het algoritme een niet-bestaand treinnummer is.

2.3.1 Verzameling van toestanden

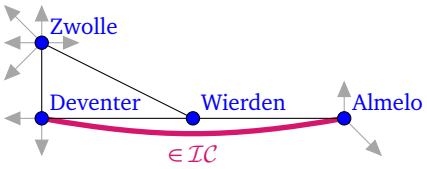
Definieer Z_k als de verzameling van toestanden Y_i in de k'de iteratie en $\mathcal{Z}$ als de verzameling van alle Z_k's:

$$Z_k = \{Y_0, Y_1, Y_2, \cdots\};$$
$$\mathcal{Z} = \{Z_0, Z_1, Z_2, \cdots\}.$$

In het algoritme zal telkens uit de laatste verzameling Z_k de nieuwe verzameling Z_{k+1} worden gegenereerd. De eerdere verzamelingen van toestanden $Z_0, \ldots, Z_{k-1}$ zullen niet meer worden gebruikt bij het genereren van nieuwe toestanden. In hoofdstuk 3 zal hier verder op worden ingegaan.

2.3.2 Intercity-trajecten

Er zijn trajecten waar Intercity's rijden die niet op alle stations stoppen, in het bijzonder Intercity's die niet op bepaalde knooppunten in het netwerk stoppen. Voor elk traject waar dit het geval is, maken we een extra lijn aan in de graaf, zie figuur A.3. De verzameling van deze specifieke lijnen noemen we $\mathcal{IC}$.
Om het beeld wat duidelijker te maken, staat hiernaast in figuur 2.2 een voorbeeld voor het Intercity-traject Deventer–Almelo. Ondanks dat Wierden een knooppunt is, stoppen deze treinen niet in Wierden, maar rijden wel over de twee deeltrajecten Deventer–Wierden en Wierden–Almelo. De dikke roze lijn geeft het Intercity-traject aan.



Figuur 2.2: *Voorbeeld van een Intercity-traject.*

Voor elke lijn $\tilde{e} \in \mathcal{IC}$ bepalen we de 'deeltrajecten' $e_1, e_2, \ldots, e_\xi \in E(G)$ waarvoor het programma zal controleren of deze trajecten al gepasseerd zijn. Dan worden de afstanden van de niet-afgelegde deeltrajecten gesommeerd en gedefinieerd als de afstand van lijn $\tilde{e} \in \mathcal{IC}$. In formulevorm wordt dit:

$$d(\tilde{e}) = \sum_{k=1}^{\xi} d(e_k)\mathbb{1}(e_k, \mathcal{R}_i),$$

waarbij $\mathbb{1}(e_k, \mathcal{R}_i)$ gedefinieerd is als in vergelijking (2.1) met $\mathcal{R}_i$ de reeds afgelegde route.

2.3.3 Overheersende toestanden

Stel dat we twee toestanden

$$Y_1 = (x_1, s_1, v_1, \mathcal{R}_1, J_1) \text{ en}$$
$$Y_2 = (x_2, s_2, v_2, \mathcal{R}_2, J_2)$$

hebben. Als $x_1 \geq x_2$, $s_1 \leq s_2$, $v_1 = v_2$ en $\mathcal{R}_1 \subseteq \mathcal{R}_2$, dan is toestand Y_2 sowieso beter dan toestand Y_1. Anders gezegd: Y_2 overheerst Y_1. Dit betekent dat toestand Y_1 verwijderd kan worden uit de toestandenverzameling, aangezien Y_1 nooit een betere oplossing kan worden dan Y_2.

Hoofdstuk 3

Het algoritme

In hoofdstuk 2 zijn de bouwstenen geïntroduceerd die we in het algoritme nodig zullen hebben. In dit hoofdstuk zal het algoritme stap voor stap worden opgebouwd en toegelicht.

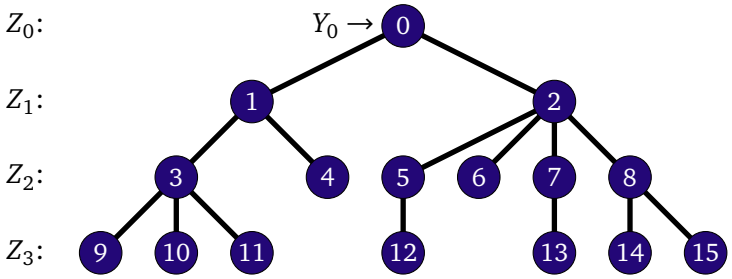
3.1 Breadth-First Search

Het algoritme is gebaseerd op het Breadth-First Search-principe (BFS) [Ski08]. Dit wil zeggen dat er per iteratie naar alle toestanden wordt gekeken en per iteratie toestanden worden geselecteerd. In elke iteratie zullen de toestanden één voor één worden bekeken en worden geanalyseerd voordat er naar de volgende iteratie wordt overgegaan, zie de volgorde in figuur 3.1.

Pseudo-code 3.1. Het algoritme om de toestandenboom te creëren en te doorzoeken ziet er als volgt uit.

```
 1: procedure BFS(Y_0)
 2:     Z_0 ← {Y_0}
 3:     k ← 0
 4:     E ← ∅                          ▷ Verzameling van 'eindpunten' in de toestandenboom.
 5:     while Z_k ≠ ∅ do
 6:         i ← 1                      ▷ Begin bij de eerste toestand in Z_k.
 7:         Z_{k+1} ← ∅                ▷ Startwaarde voor de toestandenverzameling van de nieuwe iteratie.
 8:         while i ≤ |Z_k| do
 9:             Y ← GenereerToestanden(Y_i)        ▷ Zie pseudo-code 3.3.
10:             if Y = ∅ then
11:                 E ← E ∪ Y_i        ▷ Vergelijk Y_i later met de andere eindpunten in de boom.
12:             else
13:                 Z_{k+1} ← Z_{k+1} ∪ Y          ▷ Vul de nieuwe iteratie aan.
14:             end if
15:             i ← i + 1
16:         end while
17:         k ← k + 1
18:     end while
19:     BepaalOptimum(E)                   ▷ Zie pseudo-code 3.2.
20: end procedure                          ◇
```

In het voorbeeld van figuur 3.1 is te zien dat enkele toestanden, zoals toestanden 4 en 6, geen vervolgtoestanden hebben. Deze toestanden zijn een potentiële beste oplossing. Er geldt namelijk dat de beste oplossing een 'doodlopende' toestand moet zijn, dus een toestand zonder vervolgtoestanden. Dit is eenvoudig te beredeneren, want stel namelijk dat de beste oplossing, zeg Y_b, wel een vervolgtoestand Y_v heeft. Er geldt dat in toestand Y_v minstens net zo veel afstand is afgelegd als in toestand Y_b, want de route van Y_b is ook door Y_v afgelegd, dus is Y_b geen beste oplossing.



Figuur 3.1: *De toestandenboom van Breadth-First Search. De toestanden worden in oplopende volgorde van nummering doorlopen.*

Pseudo-code 3.2. Het algoritme om de beste oplossing te selecteren uit een verzameling $\mathcal{Y}$ van toestanden ziet er als volgt uit.

```
1:  procedure BepaalOptimum(𝒴)          ▷ 𝒴 = {Y_0, Y_1, Y_2, ...}, Y_i = (x_i, s_i, v_i, ℛ_i, J_i).
2:      s_max ← 0                        ▷ Startwaarde voor de hoogste afstand.
3:      for Y_i ∈ 𝒴 do
4:          if s_i ≥ s_max then
5:              s_max ← s_i              ▷ Update de hoogste afstand.
6:              Y_best ← Y_i
7:          end if
8:      end for
9:      return Y_best
10: end procedure                        ◇
```

3.2 Maximale snelheid en streefafstand

In het algoritme wordt gebruik gemaakt van twee parameters die invloed hebben op de selectie van de toestanden per iteratie en daarmee dus op de uitvoertijd en indirect ook op de prestaties van het algoritme. Deze twee parameters zijn de maximale snelheid $V_{\max}$ van een trein in de dienstregeling en de streefafstand M. Deze M is een afstand waarvan verwacht of gehoopt wordt dat deze zeker haalbaar is binnen het gegeven tijdsbestek.

Bij een zekere toestand $Y = (x, s, v, \mathcal{R}, J)$ wordt er gekeken naar de resterende tijd $(T - x)$ en de tot dan toe bereikte afstand s. De maximaal te behalen afstand vanaf deze toestand is

$$s + V_{\max} \cdot (T - x).$$

Wanneer $s + V_{\max} \cdot (T - x) < M$, dan betekent dat dat de streefafstand niet meer te halen is binnen het tijdsbestek, dus kan deze toestand worden verwijderd uit de toestandenverzameling Z_k.

Wanneer er een toestand wordt bereikt waarbij $s > M$, dan wordt de waarde van M vervangen door de waarde van s en wordt daarna met de nieuwe M doorgerekend.

Pseudo-code 3.3. Het algoritme om de nieuwe toestanden te genereren uit een toestand $Y = (x, s, v, \mathcal{R}, J)$ ziet er als volgt uit.

```
1:  procedure GENEREERTOESTANDEN(Y_0)                ▷ Y_0 = (x_0, s_0, v_0, R_0, J_0).
2:      Y ← ∅                                        ▷ Startwaarde voor Y.
3:      B ← BUURTREINEN(x_0, v_0, J_0)               ▷ B = {B_1, B_2, ...} met B_i = (x_i, v_i, j_i, e_i).
                                                     ▷ j_i = treinnummer, e_i = trajectnummer.
4:      for B_i ∈ B do
5:          R_i ← R_0
6:          s_i ← s_0
7:          controlevariabele ← 0
8:          if e_i ∈ IC then
9:              E ← {alle deeltrajecten}             ▷ Zie paragraaf 2.3.2, E = {E_1, E_2, ..., E_ξ}.
10:         else
11:             E ← {e_i}
12:         end if
13:         for E_k ∈ E do
14:             if E_k ∉ R_i then
15:                 s_i ← s_i + d(E_k)               ▷ d(E_k) is de afstand van traject E_k.
16:                 controlevariabele ← 1
17:             end if
18:             R_i ← R_i ∪ E_k
19:         end for
20:         if controlevariabele = 1 then
21:             Y ← Y ∪ {(x_i, s_i, v_i, R_i, j_i)}
22:         end if
23:     end for
24:     return Y
25: end procedure                                    ◇
```

De waarden van $V_{\max}$ en M hebben grote invloed op de resultaten, dus deze parameters moeten slim gekozen worden. In tabel 4.1 staan een aantal resultaten waarbij $V_{\max}$ en M gevarieerd worden. Hierin is duidelijk te zien dat de parameters een groot verschil in resultaat kunnen maken.

3.3 Extra aannames

Naast de aannames die in paragraaf 2.2 worden gedaan, doen we in deze paragraaf nog enkele extra aannames om de uitvoertijd van het programma te verkorten.

Aanname 3.4. Elk traject mag maximaal één keer worden afgelegd, met uitzondering van trajecten korter dan vijf kilometer. Trajecten korter dan vijf kilometer mogen twee keer worden afgelegd, maar tellen qua afstand wel maximaal één keer mee. ◇

Aanname 3.5. Een overstap mag niet langer dan 31 minuten duren. Een toestand die een overstap van 32 minuten of meer bevat, zal worden verwijderd uit de toestandenverzameling. ◇

Hoofdstuk 4

Resultaten

In dit hoofdstuk zullen de resultaten van het algoritme worden besproken. Allereerst beginnen we met een klein voorbeeld waarbij we de parameters v_0, $V_{\max}$ en M variëren en bekijken we hoeveel invloed deze parameters hebben.

4.1 Variatie in parameters

Om de verschillen in de resultaten duidelijk te krijgen, laten we het algoritme lopen voor de beginstations met de laagste en de hoogste graad, namelijk Glanerbrug (graad 1) en Utrecht Centraal (graad 12). Omdat dit slechts een voorbeeld betreft, laten we de simulatie niet over 24 uur lopen, maar slechts van 6:00 uur tot 12:00 uur. De resultaten staan in onderstaande tabel.

Tabel 4.1: *Enkele resultaten met verschillende parameters (simulatie van 6:00 uur tot 12:00 uur).*

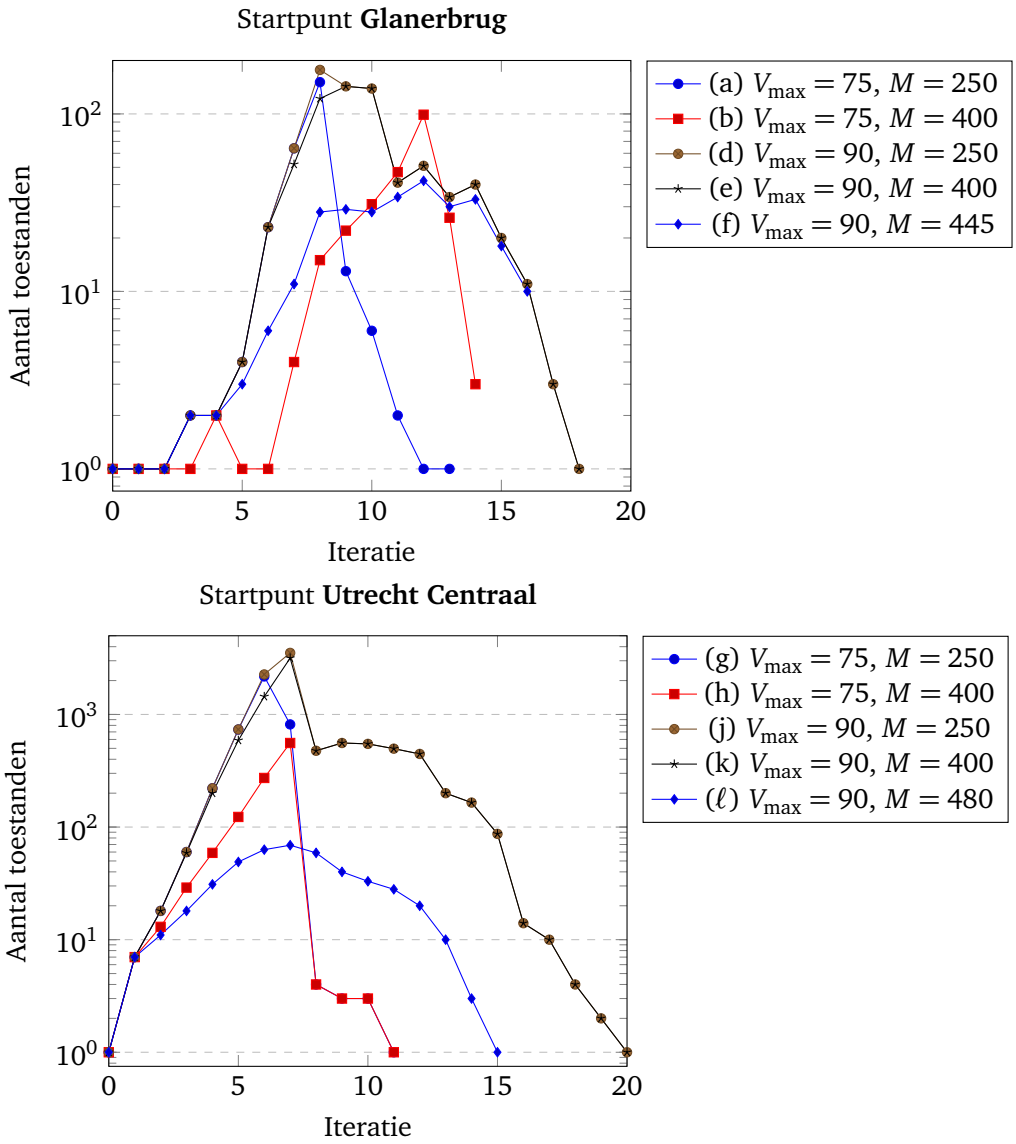
	Beginstation v_0	$V_{\max}$ (km/u)	M (km)	Resultaat (km)	Uitvoertijd (m:ss)
(a)	Glanerbrug	75	250	444.5	0:07
(b)	Glanerbrug	75	400	443.8	0:07
(c)	Glanerbrug	75	440	5.7	$<$ 0:01
(d)	Glanerbrug	90	250	448.6	0:19
(e)	Glanerbrug	90	400	448.6	0:18
(f)	Glanerbrug	90	445	448.6	0:07
(g)	Utrecht Centraal	75	250	461.8	1:40
(h)	Utrecht Centraal	75	400	461.8	0:29
(i)	Utrecht Centraal	75	460	40.0	$<$ 0:01
(j)	Utrecht Centraal	90	250	484.3	4:11
(k)	Utrecht Centraal	90	400	484.3	3:28
(ℓ)	Utrecht Centraal	90	480	418.5	0:12

Te zien is dat een hogere $V_{\max}$ een hoger resultaat geeft. Dit komt omdat er bij een hoge $V_{\max}$ minder wordt weggegooid dan bij een lage $V_{\max}$. Hetzelfde geldt voor simulaties met een lage M ten opzichte van die met een hoge M.
Het risico bestaat dat bij een lage $V_{\max}$ of hoge M de optimale oplossing wordt weggegooid. Dit heeft te maken met de mogelijkheid dat in de optimale oplossing het eerste deel van de oplossing een lagere gemiddelde snelheid heeft dan het laatste deel van de oplossing. Dit is te zien bij

simulatie (ℓ) in tabel 4.1 ten opzichte van simulaties (j) en (k). Doordat de streefafstand op 480 kilometer wordt gezet, wordt de oplossing van 484.3 kilometer in een vroeg stadium weggegooid, waardoor de hoogste oplossing van (ℓ) onder de 420 kilometer blijft.

4.2 Aantal toestanden per iteratie

Om te laten zien hoeveel geheugen het programma van een computer vraagt, zetten we in figuur 4.1 het aantal toestanden uit tegen het nummer van de iteratie. We nemen dezelfde simulaties als in de vorige paragraaf.

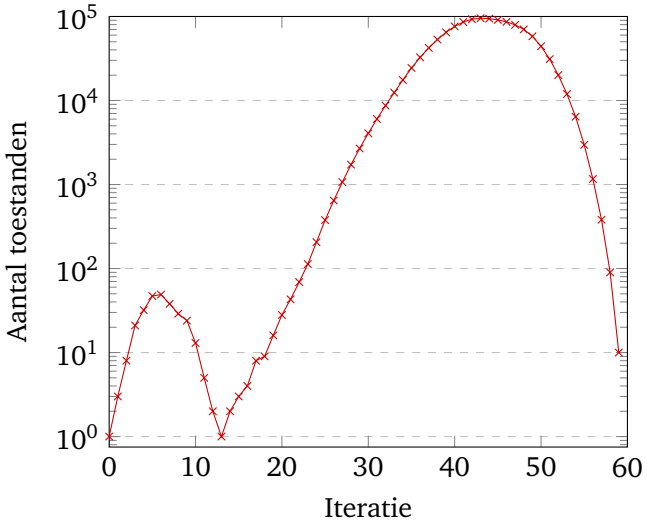


Figuur 4.1: *Het aantal iteraties van de simulaties uit tabel 4.1 met startpunt Glanerbrug respectievelijk Utrecht Centraal.*

In figuur 4.1 is te zien dat de simulaties met startpunt Utrecht Centraal een piek hebben die

ongeveer 100 keer zo hoog is dan de piek van de simulaties met startpunt Glanerbrug. Bovendien stijgen de grafieken van Utrecht Centraal direct na de eerste iteratie, terwijl die van Glanerbrug pas na de derde iteratie begint te stijgen en minder snel dan de grafieken van Utrecht Centraal. Dit is te verklaren doordat Glanerbrug op één van de uitlopers van de graaf zit, terwijl Utrecht Centraal ongeveer in het midden van de graaf zit en dus veel meer routemogelijkheden heeft.

Wanneer we naar simulaties over 24 uur kijken, zien we een interessant verloop van het aantal toestanden.



Figuur 4.2: *Het aantal toestanden per iteratie bij de 24-uurssimulatie met beginstation Alphen aan den Rijn.*

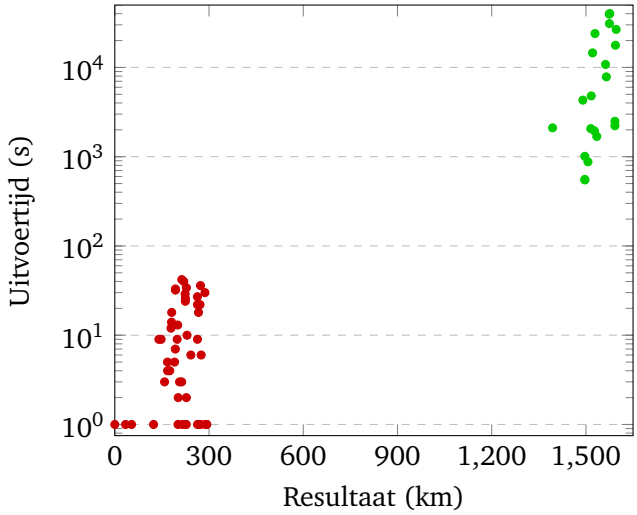
Te zien is dat er bij de dertiende iteratie een dip zit in de grafiek. Dit is te verklaren omdat op dit punt ongeveer het einde van het nachtnet zit. In het nachtnet rijden er namelijk treinen op een beperkt aantal trajecten. Veel van de toestanden rond de vijfde iteratie komen daarom op een gegeven moment op een punt uit waar men heel lang moet wachten tot er weer een trein vertrekt en dus worden deze toestanden uit de toestandenverzameling verwijderd.

4.3 Resultaten 24-uurssimulaties

De onderzoeksvraag was hoeveel afstand er binnen 24 uur kon worden afgelegd, dus laten we eens kijken naar simulaties over 24 uur. In deze paragraaf zijn alle simulaties uitgevoerd met de parameters $V_{\max} = 70$ km/u en $M = 1600$ kilometer voor een gemiddelde zaterdagdienstregeling in 2016. Door beperkingen in tijd en rekenkracht is het helaas niet gelukt om simulaties met andere parameters te doen.

In figuur 4.3 staan de resultaten van elk startstation uitgezet tegen de uitvoertijd en het resultaat is opmerkelijk. Te zien is dat er twee groepen resultaten ontstaan, namelijk de resultaten van de stations waar snel duidelijk wordt dat de streefafstand bij lange na niet gehaald wordt (de rode punten) en de resultaten die heel ver komen (de groene punten). De rode simulaties zijn over het algemeen binnen 45 seconden klaar, aangezien deze aan het begin al niet ver komen en dus vrij snel opgemerkt wordt dat de streefafstand niet gehaald wordt. De uitvoertijd van de groene

simulaties fluctueert wat meer. Afhankelijk van het startstation duurt een simulatie tussen de negen minuten en elf uur.



Figuur 4.3: *De resultaten van alle startstations uitgezet tegen de uitvoertijd.*

In figuur A.2 staat de spoorkaart weergegeven met per station het resultaat. Aangezien de resultaten daar verdeeld zijn in categorieën en dus niet zo exact zijn, staan in tabel 4.2 de vijf beste resultaten, de 'groene' resultaten[1] met de kortste en langste uitvoertijd en, we zitten immers in Twente, de resultaten van de twee grootste Twentse stations.

Tabel 4.2: *Enkele resultaten van de 24-uurssimulaties.*

#	**Beginstation**	**Resultaat (km)**	**Uitvoertijd (u:mm:ss)**
1	Alphen aan den Rijn	1596.3	7:24:28
2	Gouda	1594.0	4:55:24
3	Elst	1592.3	0:41:33
3	Nijmegen	1592.3	0:37:01
5	Rotterdam Centraal	1575.7	11:09:35
17	Amsterdam Bijlmer ArenA	1496.5	0:09:09
74	Enschede	52.9	< 0:00:01
79	Hengelo	45.3	< 0:00:01

Te zien is in figuur A.2 dat de route het beste kan beginnen in het midden van het land, aangezien er tussen Rotterdam en Arnhem een hele strook met groene punten ligt. Er zijn echter ook regio's die helemaal rood kleuren en waar het dus absoluut niet handig om te starten, zoals Friesland, Groningen, Zuid-Limburg en (helaas) Twente.

4.4 Testcase

Op zaterdag 18 juni 2016 heb ik zelf het programma in de praktijk getest. Van tevoren is het programma uitgevoerd voor alle beginstations in het netwerk met de parameters $V_{\max} = 70$ km/u

[1] Met een 'groen' resultaat wordt bedoeld dat deze in het groene gebied van figuur 4.3 ligt.

en $M = 1600$ kilometer. De werkzaamheden van deze dag waren in de dienstregeling opgenomen, dus daar werd rekening mee gehouden. De reis met het hoogste resultaat legde 1573.7 kilometer af, begon om 0:23 uur in Nijmegen en eindigde om 0:14 uur in Hengelo.
Helaas waren er onderweg een aantal storingen en vertragingen waardoor de planning aangepast moest worden en het programma dus een nieuwe route moest genereren. Dit verliep niet altijd even gemakkelijk. Om een voorbeeld te geven: rond 4 uur 's nachts was er een storing tussen Leiden Centraal en Schiphol Airport, waardoor op Schiphol een aansluiting gemist werd. Hierdoor moest er een nieuwe simulatie uitgevoerd worden met beginpunt Schiphol Airport, begintijd 4:30 uur en eindtijd 0:23 uur. Deze simulatie duurde echter zo lang dat er op dat moment is gekozen om de simulatie af te breken en voorlopig de oorspronkelijke route te vervolgen en op een later moment in de route een nieuwe simulatie te doen. Na de testcase is deze simulatie wel afgemaakt en bleek uiteindelijk zo'n 8.5 uur nodig te hebben om de simulatie uit te voeren. Hier zal bij de discussie in hoofdstuk 6 verder op worden ingegaan.

Hoofdstuk 5

Conclusie

In hoofdstuk 1 is de onderzoeksvraag *"Wat is de maximale afstand die in een beperkte tijdspanne per spoor in Nederland is af te leggen?"* gesteld, in het bijzonder binnen 24 uur.
In dit verslag, om precies te zijn in hoofdstukken 2 en 3, is een model omschreven die een goede benaderende oplossing geeft. Door beperkingen in tijd en rekenkracht van computers was het niet mogelijk om de optimale oplossing te vinden, maar wel een goede benadering ervan. Met behulp van het Breadth-First Search-algoritme wordt vanaf een gegeven beginstation stap voor stap gekeken hoe veel afstand er afgelegd kan worden per iteratie van het algoritme. Door slim de parameters $V_{\max}$ (maximale snelheid van een trein in de dienstregeling) en M (streefafstand) te kiezen, kan het programma een goede planning genereren waarbij binnen de gegeven tijdslimiet zo veel mogelijk afstand wordt afgelegd. Het beste is om $V_{\max}$ zo hoog mogelijk en M zo laag mogelijk te kiezen, aangezien dan het minste aantal toestanden wordt weggegooid. Echter loop je dan snel tegen de beperkingen van de computer aan.

In hoofdstuk 4 werden de resultaten van de simulaties besproken. Daaruit bleek dat de keuzes van $V_{\max}$ en M zeer bepalend zijn voor de resultaten. Ook de keuze van het beginstation maakt een groot verschil uit in resultaat. In figuur 4.3 is duidelijk te zien dat er een tweedeling ontstaat tussen beginstations die een grote afstand halen en beginstations waarvan al snel duidelijk wordt dat deze niet ver zullen komen en dat daarom deze simulaties snel worden afgebroken.

Een nadeel van het programma is dat de uitvoertijd zeer uiteenlopend is, afhankelijk van de parameters. De 'groene' resultaten uit figuur 4.3 hebben een uitvoertijd uiteenlopend van negen minuten tot elf uur. Al met al bereikt het programma zijn doel en kunnen we antwoord geven op de onderzoeksvraag. Binnen 24 uur is er per spoor in Nederland een afstand te halen van minimaal[1] 1596.3 kilometer, aldus tabel 4.2.

[1] Dit is berekend met de parameters $V_{\max} = 70$ km/u en $M = 1600$ kilometer.

Hoofdstuk 6

Discussie

In dit hoofdstuk zullen enkele onderwerpen worden behandeld waar bij later onderzoek dieper op kan worden ingegaan.

6.1 Invloed van tussenstations

Bij het opbouwen van de graaf voor dit model (figuur A.2) is er voor gekozen om alleen de belangrijkste knooppunten in de graaf te zetten, zie aanname 2.1. Alle kleine tussenstations worden dus niet verwerkt in de graaf, zoals beschreven in paragraaf 2.1.2. Nu kan men zich afvragen of dit invloed heeft op de optimale oplossing en of je dus kilometers misloopt door tussenstations niet te verwerken. Het blijkt dat dit wel degelijk een (gering) effect heeft op de oplossing; dit zal gedemonstreerd worden aan de hand van voorbeeld 6.1.

Voorbeeld 6.1 (Fictief). Stel dat het programma een planning genereert waarvoor geldt dat:

- de reis begint om 0:00 uur op een willekeurig station en eindigt om 23:45 uur op station Hengelo;
- de reis niet het traject Hengelo–Enschede bevat.

Stel verder dat er een trein rijdt met de volgende dienstregeling:

km	*station*	*tijd*
0	Hengelo	23:52
3.8	Enschede Kennispark	23:57
7.6	Enschede	0:02

Deze trein wordt niet als optie gegeven, aangezien de trein na 0:00 uur in Enschede aankomt. Echter kan je binnen de tijd nog wel tot Enschede Kennispark reizen, maar omdat dit station niet als punt in de graaf zit, wordt de mogelijkheid om in Enschede Kennispark te eindigen niet weergegeven. Dit betekent dus een verlies van 3.8 kilometer. ◇

Nu zullen we het worstcasescenario bekijken, oftewel: wat is de maximale afstand die kan worden misgelopen door de tussenstations uit de graaf te verwijderen (aanname 2.1)?

Voorbeeld 6.2 (Worstcasescenario). Het langste traject in de graaf is het traject Roosendaal–Vlissingen (75.5 kilometer). Stel nu dat je een planning hebt waarbij je vanuit Roosendaal niet

meer binnen de tijd in Vlissingen kunt komen, maar nog wel in Vlissingen Souburg (laatste station voor Vlissingen, zie figuur 6.1).

Figuur 6.1: *Route Roosendaal–Vlissingen met alle tussenstations. [NS16a]*

De afstand Roosendaal–Vlissingen Souburg is 73.6 kilometer en aangezien Vlissingen Souburg en alle andere tussengelegen stations niet als punten in de graaf zitten, loop je deze afstand mis. Deze 73.6 kilometer is dus ongeveer de langste afstand die je zou kunnen mislopen door aanname 2.1, maar de kans dat dit gebeurt is relatief klein. ◇

6.2 Het programma uitvoeren tijdens de reis

In principe was het idee om het programma zodanig te ontwerpen dat het programma ook tijdens de reis, in geval van een storing of vertraging, kan worden uitgevoerd om een nieuwe route te bepalen. Echter is het programma vooralsnog te traag om onderweg fatsoenlijk te worden uitgevoerd. Een voorbeeld hiervan is in paragraaf 4.4 behandeld. Later onderzoek zou het programma sneller kunnen maken om het wel te kunnen uitvoeren tijdens een reis. Het algoritme zou eventueel verbeterd kunnen worden of het programma zou efficiënter kunnen worden geprogrammeerd.

6.3 Niet meer gebruikte toestanden verwijderen uit geheugen

In het programma wat voor deze opdracht geschreven is, is er voor gekozen om alle toestanden die gegenereerd zijn, oftewel de complete verzameling $\mathcal{Z}$ uit paragraaf 2.3.1, op te slaan in het computergeheugen. Helaas heeft dit meerdere malen tot een `MemoryError` geleid. Voor verder onderzoek wordt aanbevolen om per iteratie alleen de potentiële beste oplossingen op te slaan (zie paragraaf 3.1) en de rest van de toestanden te verwijderen uit het computergeheugen.

Referenties

[NS16a] Nederlandse Spoorwegen. Fragment uit vertrekstaat 1 van Roosendaal, 2016. URL: http://www.ns.nl/vertrekstaten/rsd1.pdf, geraadpleegd op 13 juni 2016.

[NS16b] Nederlandse Spoorwegen. Spoorkaart van Nederland, 2016. URL: http://www.ns.nl/binaries/_ht_1448541342743/content/assets/ns-nl/dienstregeling/spoorkaart-2016-met-trajecten-1-t-m-128.pdf, geraadpleegd op 16 mei 2016.

[Ski08] Steven S. Skiena. *The Algorithm Design Manual*. Springer, tweede editie, 2008. Blz. 162–169.

Bijlage A

Tabellen en figuren

A.1 Lijst met variabelen

Hieronder, in tabel A.1, vindt u een overzicht van alle variabelen die in dit verslag worden gebruikt en de pagina waarop ze worden geïntroduceerd.
De letters i en k worden als indices gebruikt.

Tabel A.1: *Lijst met variabelen.*

Symbool	Omschrijving	Pagina
$\mathbb{1}(e_i, \mathcal{R}_{i-1})$	Indicatorfunctie met betrekking tot het passeren van lijn e_i, gedefinieerd zoals in vergelijking (2.1).	3
$\alpha_v(j)$	Vertrektijd van trein j vanuit punt v.	3
$\beta_w(j)$	Aankomsttijd van trein j in punt w.	3
$D_{\vec{vw}}$	Dienstregeling van station v naar station w.	4
$d(v,w)$	Afstand (in kilometers) van punt v naar punt w. Er geldt dat $d(v,w) = d(w,v) \geq 0$.	1
E	Verzameling van spoorlijnen tussen de stations.	1
e_i	Lijn (traject) in de graaf, ook wel aangeduid als $\vec{vw}$.	3
$G(V,E)$	Graaf van het spoornetwerk (figuur A.2).	1
$\mathcal{IC}$	Verzameling van Intercity-trajecten (figuur A.3).	6
J_i	Treinnummer van de laatste trein in route $\mathcal{R}_i$.	5
$j(v,w)$	Treinnummer van een trein van punt v naar punt w.	3
M	Streefafstand.	8
$\mathcal{R}_i$	Afgelegde route tot stap i.	3
s_i	Afgelegde afstand tot stap i.	5
T	Tijdslimiet (> 0).	3
$\mathcal{T}_{\vec{vw}}$	Verzameling van treinen van punt v naar punt w.	4
$t_{\vec{vw}}(x)$	Totale reisduur (> 0) vanaf punt v naar punt w vanaf tijdstip x.	3
V	Verzameling van knooppuntstations.	1
$V_{\max}$	Maximale snelheid van een trein in de dienstregeling.	8
v, w	Punten (stations) in de graaf.	3
$\vec{vw}$	Lijn (traject) in de graaf, ook wel aangeduid als e_i.	3
x_i	Verstreken tijd na stap i.	3
x_0	Begintijdstip.	4

Vervolg op volgende bladzijde

Tabel A.1 – *Vervolg van vorige bladzijde*

Symbool	Omschrijving	Pagina
Y_i	Toestand in de vorm $Y_i = (x_i, s_i, v_i, \mathcal{R}_i, J_i)$.	5
Z_k	Verzameling van toestanden in de k'de iteratie.	6
$\mathcal{Z}$	Verzameling van toestandenverzamelingen Z_k.	6

A.2 Lijst van stations

Tabel A.2: *Lijst van stations, alfabetisch gesorteerd op afkorting.*

Afkorting	Station
Ah	Arnhem Centraal
Ahp	Arnhem Velperpoort
Alm	Almere Centrum
Amf	Amersfoort
Aml	Almelo
Ampo	Almere Poort
Amr	Alkmaar
Apd	Apeldoorn
Apn	Alphen aan den Rijn
Asb	Amsterdam Bijlmer ArenA
Asd	Amsterdam Centraal
Asdm	Amsterdam Muiderpoort
Asdz	Amsterdam Zuid
Ass	Amsterdam Sloterdijk
Bd	Breda
Bdpb	Breda-Prinsenbeek
Bkl	Breukelen
Br	Blerick
Brn	Baarn
Btl	Boxtel
Db	Driebergen-Zeist
Ddr	Dordrecht
Dld	Den Dolder
Dtc	Doetinchem
Dv	Deventer
Dvd	Duivendrecht
Dz	Delfzijl
Ed	Ede-Wageningen
Edn	Eijsden
Eghm	Eygelshoven Markt
Ehv	Eindhoven
Ekz	Enkhuizen
Emn	Emmen
Es	Enschede

Vervolg op volgende bladzijde

Tabel A.2 – *Vervolg van vorige bladzijde*

Afkorting	**Station**
Est	Elst
Gbr	Glanerbrug
Gd	Gouda
Gdm	Geldermalsen
Gerp	Groningen Europapark
Gn	Groningen
Gr	Gorinchem
Gv	Den Haag HS
Gvc	Den Haag Centraal
Hdr	Den Helder
Hfd	Hoofddorp
Hgl	Hengelo
Hld	Hoek van Holland Haven
Hlds	Hoek van Holland Strand
Hlgh	Harlingen Haven
Hlm	Haarlem
Hn	Hoorn
Hrl	Heerlen
Ht	's-Hertogenbosch
Hvl	Hoevelaken
Hvs	Hilversum
Hwd	Heerhugowaard
Kpn	Kampen
Krd	Kerkrade Centrum
Laa	Den Haag Laan van NOI
Ledn	Leiden Centraal
Lg	Landgraaf
Lls	Lelystad Centrum
Lw	Leeuwarden
Mp	Meppel
Mrb	Mariënberg
Mrn	Maarn
Msw	Maassluis West
Mt	Maastricht
Mtr	Maastricht Randwyck
Ndb	Naarden-Bussum
Nm	Nijmegen
Nsch	Bad Nieuweschans
Odz	Oldenzaal
Rai	Amsterdam RAI
Rd	Roodeschool
Rhn	Rhenen
Rlb	Rotterdam Lombardijen
Rm	Roermond

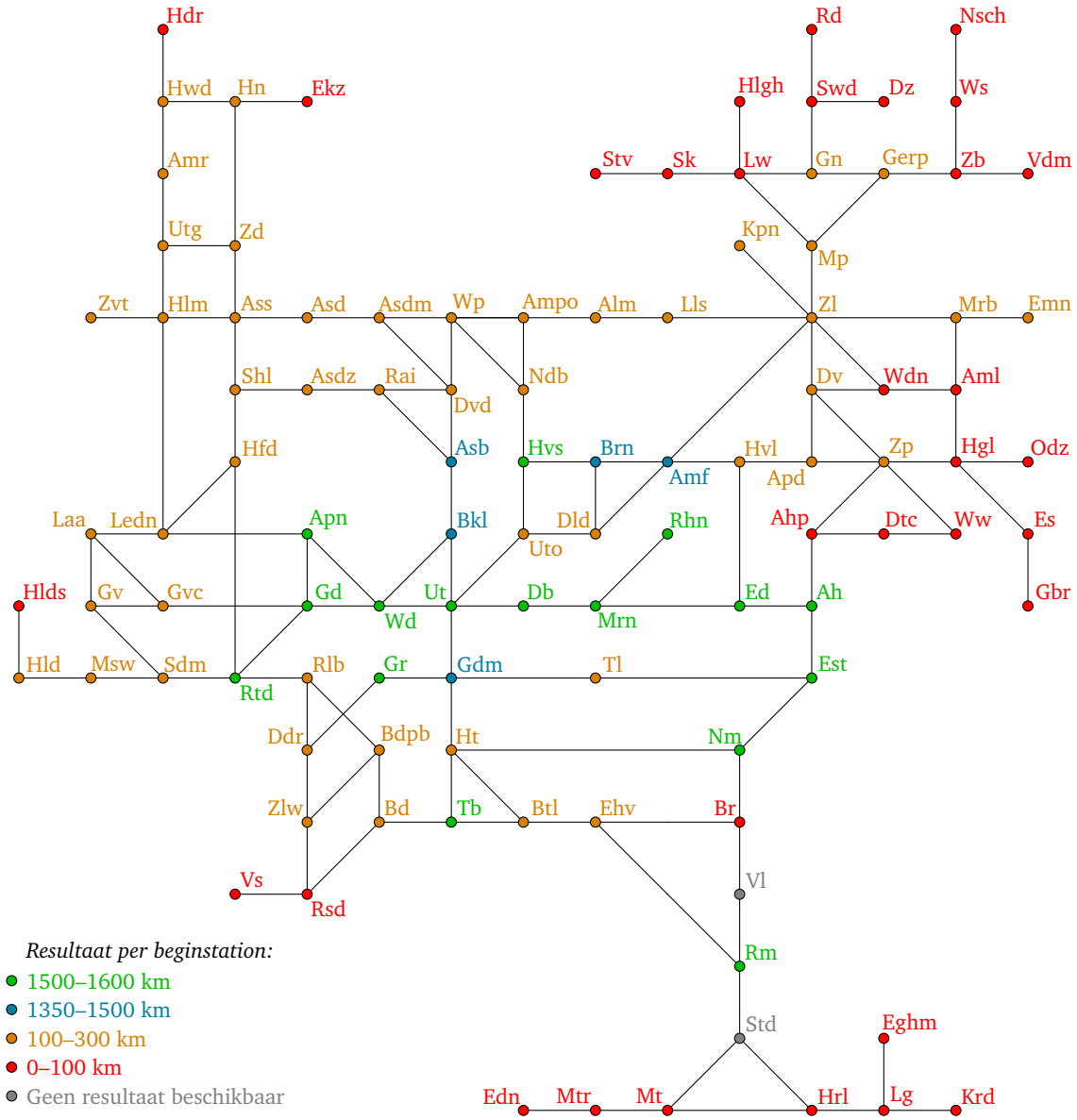
Vervolg op volgende bladzijde

Tabel A.2 – *Vervolg van vorige bladzijde*

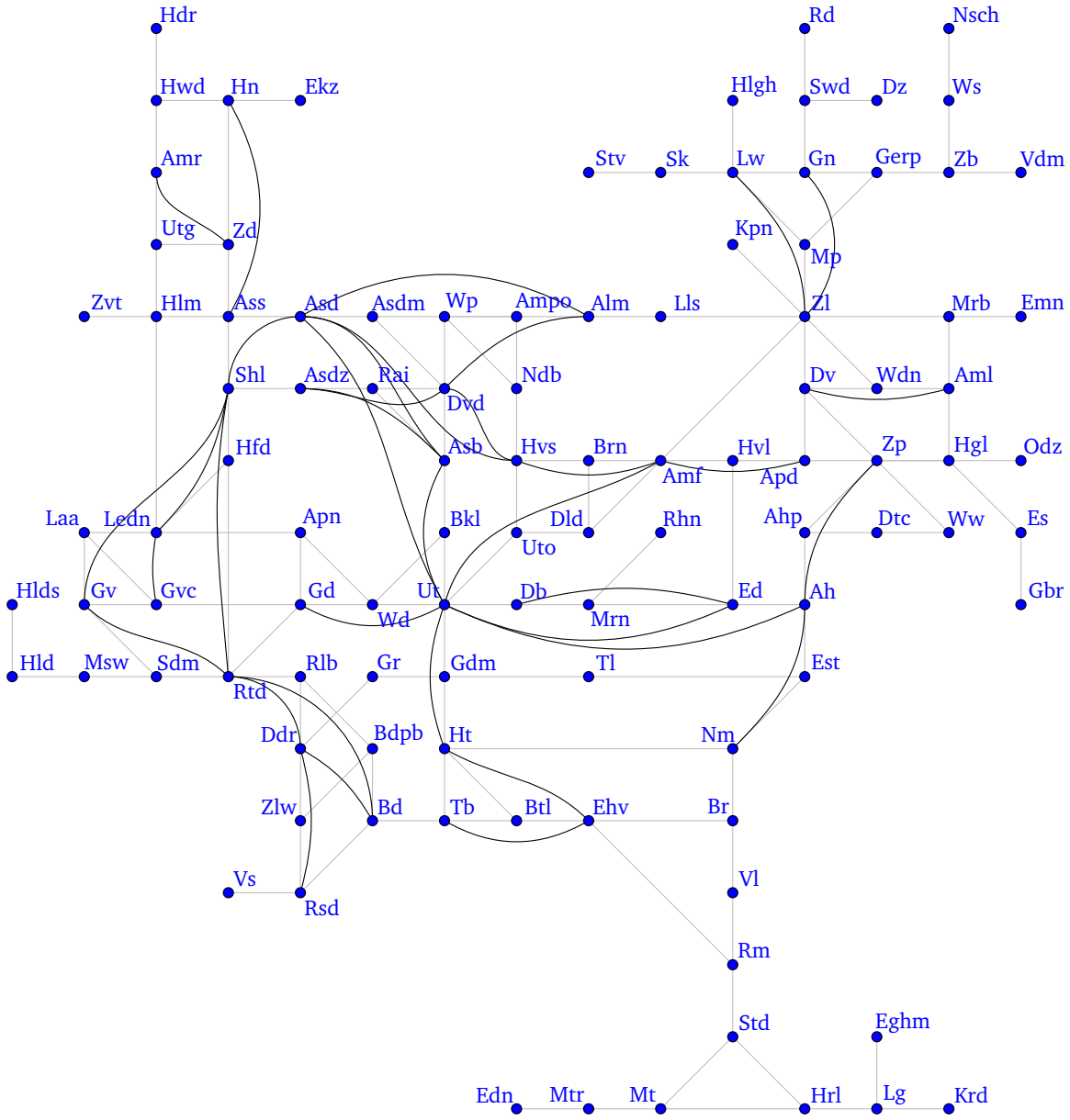
Afkorting	**Station**
Rsd	Roosendaal
Rtd	Rotterdam Centraal
Sdm	Schiedam Centrum
Shl	Schiphol Airport
Sk	Sneek
Std	Sittard
Stv	Stavoren
Swd	Sauwerd
Tb	Tilburg
Tl	Tiel
Ut	Utrecht Centraal
Utg	Uitgeest
Uto	Utrecht Overvecht
Vdm	Veendam
Vl	Venlo
Vs	Vlissingen
Wd	Woerden
Wdn	Wierden
Wp	Weesp
Ws	Winschoten
Ww	Winterswijk
Zb	Zuidbroek
Zd	Zaandam
Zl	Zwolle
Zlw	Lage Zwaluwe
Zp	Zutphen
Zvt	Zandvoort aan Zee

A.3 Spoorkaart

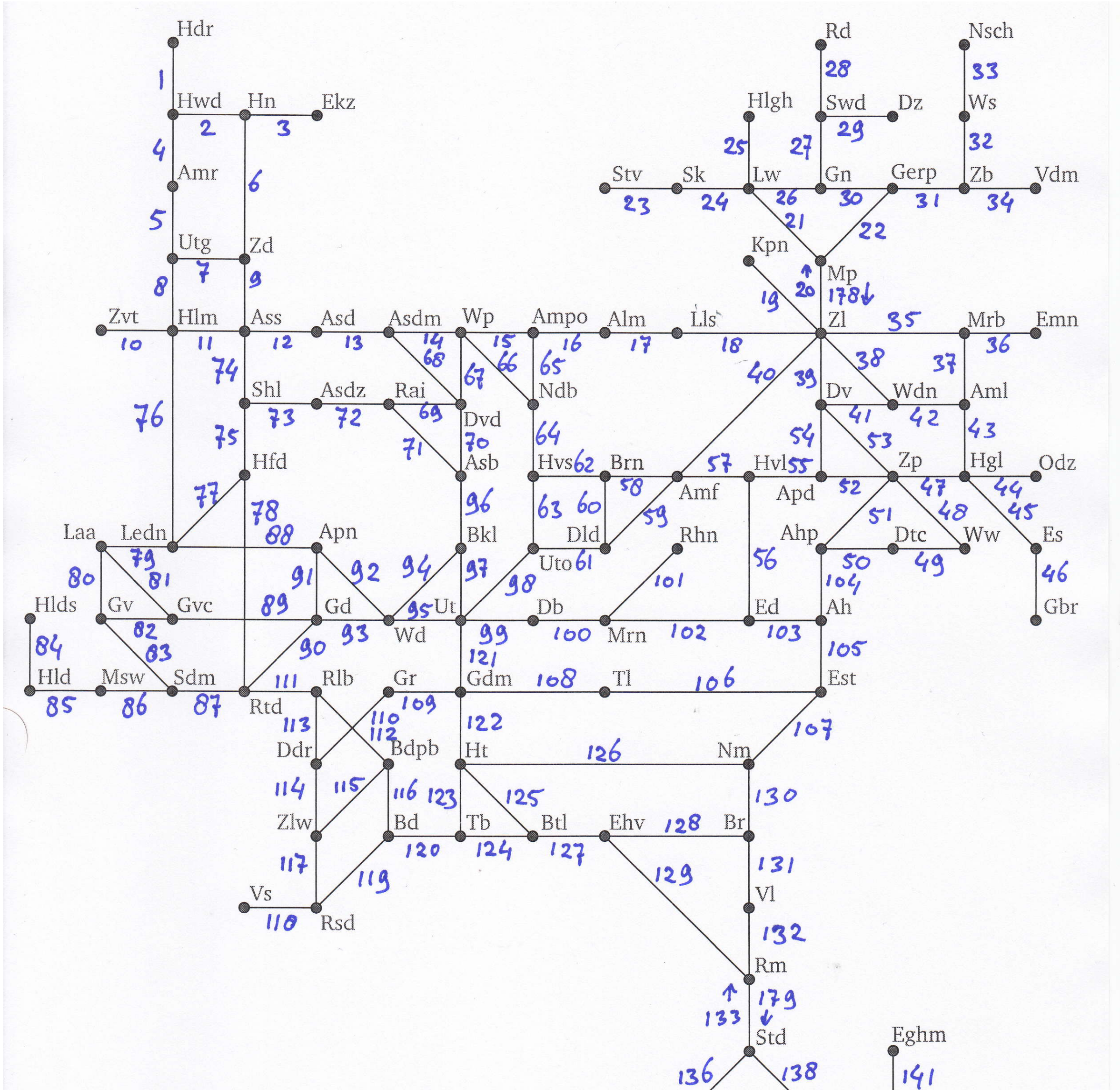
Figuur A.1: *Spoorkaart van Nederland, dienstregeling 2016.* [NS16b]



Figuur A.2: *Spoorkaart van Nederland, weergegeven in een graaf. De kleuren geven per punt aan wat het resultaat is als deze als beginpunt wordt gekozen. Hierbij zijn* $V_{\max} = 70$ *km/u en* $M = 1600$ *km.*



Figuur A.3: *De Intercity-trajecten in het netwerk. De grijze lijnen zijn van de oorspronkelijke graaf G, de zwarte lijnen zitten in de verzameling $\mathcal{IC}$.*



Figuur A.4: *De spoorkaart van figuur A.2 met bijbehorende trajectnummers.*
N.B.: Bij de twee trajecten Zwolle–Meppel (Zl–Mp) en Roermond–Sittard (Rm–Std) staan twee trajectnummers. Deze trajecten mogen namelijk twee keer gepasseerd worden, dus zijn deze trajecten in de dienstregeling opgesplitst in twee trajecten, namelijk 20 respectievelijk 133 met de treinen in noordelijke richting en 178 respectievelijk 179 met de treinen in zuidelijke richting.

Bijlage B

Handleiding voor het programma

In dit hoofdstuk zal worden uitgelegd hoe het programma moet worden uitgevoerd.

Het programma is geschreven in de programmeertaal Python en bestaat uit vier bestanden. Deze bestanden zijn:

`BOs1089099.py`. Dit is het hoofdbestand waar alle subprogramma's worden gedefinieerd en alle benodigde packages worden opgeroepen.

`drgl2016.sql`. In dit bestand staat de database met de dienstregeling die in het hoofdbestand wordt opgeroepen. In deze database staan alle treinen die rijden tussen 0:00 uur en 4:00 uur de volgende dag, dus de dienstregeling beslaat 28 uur.

`BOmatrices.py`. Dit is een hulpbestand waarin de matrix met de afstanden tussen de knooppuntstations wordt gedefinieerd.

`BOstations.py`. Dit bestand is bedoeld voor het gemak van de gebruiker. Hierin worden variabelen gedefinieerd waarvan de naam gelijk is aan de afkorting van een knooppuntstation (in kleine letters) en de waarde gelijk is aan het volgnummer van het knooppuntstation. Als voorbeeld: Enschede heeft in de graaf volgnummer 34 gekregen, dus de variabele `es` (de afkorting van Enschede) heeft de waarde 34. Alle afkortingen zijn te vinden in tabel A.2.

Plaats bovenstaande vier bestanden in dezelfde map en open het bestand `BOs1089099.py` in de Python-compiler van uw keuze, bijvoorbeeld IDLE. Het hoofdprogramma waarmee de planning wordt gegenereerd heet `reis` en wordt op de onderstaande manier opgeroepen.

```
>>> reis(beginstation,begintijd,eindtijd,V_max,streefafstand,R_0)
```

De input R_0 is de reeds afgelegde route op het begintijdstip in de vorm van een lijst met trajectnummers. Deze input is optioneel en heeft als standaardwaarde de lege lijst. De trajectnummers zijn te vinden in figuur A.4.

Voorbeeld B.1. Als men wil aangeven dat men de trajecten Hengelo–Enschede (traject 45) en Enschede–Glanerbrug (traject 46) al heeft gereisd, wordt voor R_0 de waarde `[45,46]` ingevuld. ◇

De inputs `begintijd` en `eindtijd` dienen als deel van de dag te worden ingevoerd. Als hulpmiddel is er een functie `tijd` gemaakt die de tijd in uren en minuten omzet in de tijd in het deel van de dag.

Voorbeeld B.2 (functie `tijd`). Om 10:48 uur is $\frac{9}{20}$ deel van de dag voorbij, dus krijgen we onderstaande uitkomst.

```
>>> tijd(10,48)        # in de vorm: tijd(uren,minuten)
0.45
```

De functie `tijd` heeft een extra optie om het aantal dagen in te voeren als derde input. Onderstaande commando's zijn equivalent en geven allemaal het antwoord op de vierde regel.

```
1 >>> tijd(10,48,1)      # in de vorm: tijd(uren,minuten,dagen)
2 >>> tijd(34,48)        # 24 bij het aantal uren opgeteld
3 >>> 1 + tijd(10,48)    # handmatig een dag bij de tijd opgeteld
4 1.45                                                          ◇
```

Voorbeeld B.3 (functie `reis`). Als voorbeeld bekijken we hoe ver we kunnen komen als we vanuit Hengelo (`hgl`) vertrekken en van 6:00 uur tot 9:00 uur de tijd hebben. We kiezen $V_{\max} = 80$ km/u en de streefafstand is 200 kilometer. Als output krijgen we de totaal afgelegde afstand en de bijbehorende reisplanning. Wanneer er regel verschijnt met een pijltje en een nummer, dan betekent dit dat er overgestapt moet worden. In onderstaand voorbeeld reist men vanaf Hengelo met trein 1620 tot Amersfoort met stops in Almelo, Deventer en Apeldoorn, in Amersfoort moet worden overgestapt op trein 11720 naar Utrecht Centraal en in Utrecht Centraal stapt men over op trein 825 naar Eindhoven. In totaal is er dan 213.1 kilometer afgelegd.

```
>>> reis(hgl,tijd(6,0),tijd(9,0),80,200)
Totale afstand: 213.1 km
   -> 1620
06:24 Hengelo
06:35 Almelo
06:36 Almelo
06:59 Deventer
07:02 Deventer
07:12 Apeldoorn
07:13 Apeldoorn
07:37 Amersfoort
   -> 11720
07:40 Amersfoort
07:54 Utrecht Centraal
   -> 825
08:07 Utrecht Centraal
08:35 s-Hertogenbosch
08:37 s-Hertogenbosch
08:56 Eindhoven
Goede reis!                                                     ◇
```