File, File en nog eens File

Het Verminderen van de Congestie op Snelwegen door het Introduceren van een Begrijpelijk Wiskundig Model

Stan de Haas

Begeleider: Mevr. M. Visser

December 2022

Profielwerkstuk wiskunde

vwo 6

Rijnlands Lyceum Sassenheim

Samenvatting

Verschillende bedrijven en onderzoekscentra proberen, door middel van complexe wiskundige modellen, filedrukte te verminderen. De complexiteit van verkeersmodellen is alleen hoog waardoor er een kloof ontstaat tussen de specialist en de bestuurder. Voor beide partijen leidt dit tot irritatie. Dit verhoogt de hoeveelheid file doordat de naleving van de verkeersmaatregelen lager wordt. Om de filedrukte te verminderen, moet de naleving onder bestuurders worden verhoogd.

Dit onderzoek streeft naar het introduceren van een begrijpelijk wiskundig model dat de basis van verkeerstheorie uiteenzet. Hierdoor zijn bestuurders instaat eenvoudig inzicht te krijgen in de verkeersstrategieën die filedrukte verminderen. Daarnaast wordt het model gebruikt om de werking van verkeersstrategieën aan te tonen en deze met elkaar te vergelijken. Deze strategieën kunnen vervolgens worden geïmplementeerd op Nederlandse snelwegen. De onderzoeksvraag luidt als volgt: hoe kan de filedrukte op snelwegen, door middel van een voor iedereen begrijpelijk wiskundig model, worden verminderd?

Deze studie is instaat een antwoord te geven op deze onderzoeksvraag door, na het theoretisch kader en twee pilotexperimenten, een wiskundig model op te stellen. Om de validiteit van dit model te garanderen is het met vier testexperimenten gekeurd. Vervolgens zijn vier casestudies uitgevoerd. Deze zijn gebaseerd op Nederlandse verkeersknooppunten. Ook hier wordt de validiteit gewaarborgd door het uitvoeren van een herhalingsexperiment naar één van de casestudies. Uit de simulaties blijken verkeersknooppunten, al vereenvoudigd, succesvol gemodelleerd te zijn. De resultaten laten zien dat het verminderen van de filedrukte door het toepassen van verkeersstrategieën in enkele van de casestudies mogelijk is.

Het onderzoek toont aan dat de hoeveelheid congestie op snelwegen vermindert door het toepassen van intelligente verkeerssystemen zoals ramp metering en variabele snelheidslimieten. Dit is laten zien met het in dit onderzoek geïntroduceerde wiskundig model dat relatief eenvoudig is. Door de eenvoud van het model wordt ook de naleving van verkeersmaatregelen onder bestuurders verhoogd. Bestuurders krijgen immers de mogelijkheid om gemakkelijk informatie in te winnen over het vormen van file en kunnen zo de werking van verkeersstrategieën beter beseffen.

Na een bespreking van de beperkingen en de vereenvoudigingen van het wiskundige model, adviseert dit onderzoek om intelligente verkeerssystemen in hogere mate toe te passen op Nederlandse snelwegen. Daarnaast beveelt dit onderzoek het geïntroduceerde model aan. Hierdoor zijn bestuurders instaat beter inzicht krijgen in de werking van congestie. Mogelijkerwijs richt vervolgonderzoek zich op het introduceren van begrijpelijke microscopische modellen (zoals voertuigverplaatsing bij verkeerslichten) of bouwt het voort op dit onderzoek door het introduceren een complexer model die de beperkingen van dit onderzoek verhelpt.

Inhoudsopgave

Afkortingenlijst

Afkorting	**Betekenis**
a	parameter voor snelheidsberekening
A	parameter voor variabele snelheidslimieten
BFS	breadth-first search
C	gemiddelde tijdsspan van een meting
CTM	cel transmissie model
DFS	depth-first search
E	parameter voor variabele snelheidslimieten
FD	fundamenteel diagram
G	graaf
h	volgafstand
i	cel
K	dichtheid
K_{cr}	kritieke dichtheid
K_{jam}	maximale dichtheid
ΔL	lengte van een bepaald gebied
n	aantal voertuigen
N	maximaal aantal voertuigen
N_tot	totaal aantal voertuigen
NDW	Nationaal Dataportaal Wegverkeer
o	bezetting
O	complexiteit uitgedrukt in de big-O notatie
$\hat{o}$	gewenste bezetting
q	intensiteit
Q	maximale intensiteit
r	instroom vanuit een oprit
RM	ramp metering
S	volgafstand
s	uitstroom naar een afrit
t *of* k	variabele voor tijd
T_d	totaal gereisde afstand
T_s *of TTS*	totaal bestede tijd
TSP	travelling salesmen problem
ΔT	tijdsstap
v	snelheid
V	vertex
V	snelheid geïmpliceerd door VSL
v_f	vrijestroomsnelheid
v_{sms}	afstandsgemiddelde snelheid
v_{tms}	tijdsgemiddelde snelheid
VSL	variabele snelheidslimieten
w	snelheid van een shockwave
w_{max}	maximale snelheid van een shockwave
y	intensiteit per tijdstap
α	naleving
β	splitsingsverhouding
μ	aantal rijbanen
Λ	gemiddelde lengte van een voertuig

Inleiding

Paradoxaal genoeg is het de dagelijkse realiteit dat een samenleving die streeft naar efficiëntie gehinderd wordt door verkeersdrukte van en naar werk. Filedrukte roept daarom ook veel irritatie op. Zeker als het verkeer 'opeens' weer gaat rijden zonder dat de oorzaak duidelijk is geworden. In 2021 bedroeg de jaarfilezwaarte, uitgedrukt door de gemiddelde filedrukte te vermenigvuldigen met de duur van de file, dan ook 5,7 miljoen kilometerminuten. Echter, dit is nog niet eens de helft van de totale jaarfilezwaarte voor de coronapandemie (Rijkswaterstaat, 2022). Een Nederlander die minstens één keer in de week te maken heeft met file verloor in een jaar gemiddeld 37 uur (Ministerie van Infrastructuur en Waterstaat, 2020). Men gaat in die tijd liever een week op vakantie. Het verminderen van de file is om meerdere reden cruciaal. Inzichten in de werking van file door wiskundige simulaties draagt hieraan bij en verminderen zowel de irritaties van bestuurders als de filedrukte op snelwegen.

De verkeerstheorie is een nieuwe wetenschap die pas in de afgelopen decennia is opgekomen. De gevolgen ervan merkt de Nederlander, in tegenstelling tot anders wetenschappen, alleen dagelijks. Er zijn daarom veel onderzoeken gaande waardoor het een van de snelst vooruitgaande studies is. Binnen deze studie komen onderwerpen aan bod van microscopische modellen, die keuzen van individuele bestuurders onderzoeken, tot macroscopische modellen die gehele snelwegsystemen simuleren (École Polytechnique Fédérale de Lausanne, 2022). Op dit laatste is dit onderzoek gericht.

Binnen de verkeerstheorie wordt voornamelijk op optimalisering gefocust. Een aantal wetenschappelijke artikelen liggen daarom ten grondslag aan de gehele verkeerstheorie (Daganzo C. F., 1992; Daganzo C. F., 1994; Daganzo C. F., 1994; Papageorgiou, M. et al, 2003). Zij geven inzicht in verkeersmodellering en geven theoretische achtergronden. Hierdoor bouwen wetenschappelijke artikelen van vandaag de dag voort op eerder artikelen (Frejo, J. R. D., et al, 2018; Khodaker, B., et al, 2015). Als gevolg worden de artikelen steeds specifieker. Dit is ten nadelen van de begrijpbaarheid, omdat eerst onderliggende artikelen moeten worden begrepen.

Wetenschappelijke instellingen, bedrijven en overheden werken nou samen om complexe modellen te maken die de werkelijkheid zo realistische mogelijk uitbeelden. Zelfs broeikasgasuitstoot en geluidsoverlast worden in detail gemodelleerd (MIRT, 2018). Als individu zijn deze modellen echter niet meer toegankelijk. Daarentegen, er is weinig onderzoek gedaan naar het combineren van verschillende theorieën in vereenvoudigde modellen. Deze vereenvoudigde modellen geven de gemiddelde bestuurder juist meer inzicht in de werking van het verkeer. Tevens zijn bestuurders in staat om met eenvoudige modellen keuzen voor bepaalde verkeersstrategieën beter te begrijpen. Op praktisch gebied is er nog geen manier gevonden om het begrip van de bestuurder over de werking van file te verhogen en kost file voor elk veel tijd. Daarnaast zijn er veel verkeersknooppunten in Nederland, met elk hun eigen problemen. Zelfs vereenvoudigde modellen kunnen door het uitvoeren van casestudies mogelijke verbeteringen voor deze verkeerssystemen vinden.

Uit de probleemstelling valt te concluderen dat de Nederlandse bestuurder niet over de juiste tools beschikt om file en de effecten van verkeersmaatregelen te begrijpen. Hierdoor is het doel van deze studie om de Nederlander, door middel van een wiskundig model, inzicht te bieden in de werking van file en de methoden die de problemen van filedrukte aanpakken. Zo wordt ook de werking van deze methoden aangetoond. Daarnaast worden de verschillen tussen methoden zichtbaar en wordt men in staat gesteld te zien welke verkeerstrategie het beste kan worden toegepast. Daarnaast doelt deze studie op het verminderen van de filedrukte door het geïntroduceerde model toe te passen op vier casestudies.

De basis van macroscopische modellen, die in de verkeerstheorie gebruikt worden voor het modelleren van snelwegen, wordt gevormd door het cel transmissie model (Daganzo, C. F. 1992). Het is afkomstig uit 1992. Echter, het is het model dat vandaag de dag nog steeds het meest wordt gebruikt. Dit onderzoek maakt dan ook gebruik van de principes van Daganzo's cel transmissie model. Ten tweede zijn er verschillende methoden binnen de literatuur bekend om verkeersdrukte te verminderen. Er zijn daarom drie intelligentie verkeerssystemen in het model geïmplementeerd: ramp metering, variabele snelheidslimieten en het aanpassen van parameters (d.w.z. aanpassen van bijvoorbeeld het aantal rijbanen) (École Polytechnique Fédérale de Lausanne, 2022). Hierbij is gedoeld op het aanleveren van een begrijpelijk model dat tevens betrouwbare resultaten levert. De onderzoeksvraag van deze studie luidt dan ook als volgt:

'Hoe kan de filedrukte op snelwegen, door middel van een voor iedereen begrijpelijk wiskundig model, worden verminderd?'

Deze onderzoeksvraag is onderverdeeld in de volgende vijf deelvragen:

- *'Hoe functioneren grafen en de bijbehorende algoritmen op het gebied van zowel theoretische als computationele grafentheorie?'*
- *'Hoe werken de theorieën die de basis vormen van de verkeerstheorie?'*
- *'Hoe programmeert men een begrijpelijk wiskundig model dat het (Nederlandse) snelwegennet simuleert?'*
- *'Hoe kan empirische data in het wiskundige model worden verwerkt om de realiteit juist te modelleren?*
- *'Hoe wordt file in het model verminderd door het toepassen van intelligente verkeerssystemen?'*

Het belang van deze studie kan groot zijn. Indien bestuurders namelijk benul zijn van de gevolgen hun rijgedrag op de filezwaarte, neemt de naleving van intelligente verkeerssystemen toe. Naast dat de irritatie afneemt, neemt de totale filedrukte hierdoor ook af. De bestuurder wordt vandaag de dag gezien als iemand die onbegrijpelijke maatregelen dient op te volgen, terwijl vele nog steeds op eigen verstand handelen. Dit met de gevolgen van dien.

Om de onderzoeksvraag en zijn deelvragen, te beantwoorden is een wiskundig model voorgesteld die op macroscopische wijzen snelwegknooppunten met ingewikkelde onderdelen als op- en afritten weet te modelleren. Vervolgens zijn met behulp van open data van het Nationaal Dataportaal Wegverkeer vier casestudies uitgevoerd (Ministerie van Infrastructuur en Waterstaat, 2022). Deze tonen, samen met twee pilotexperiment, vier testexperimenten en één het herhalingsexperiment, de werking van het model en de werking van de verkeersstrategieën aan.

Hoofdstuk 1 presenteert een theoretisch kader waarin een overzicht van grafentheorie geven wordt. Dit vormt de basis van het computationele deel van dit onderzoek. Daarnaast komen vier verschillende bronnen over verkeerstheorie aan bod. Hoofdstuk 2 bevat de methodologie waar de keuze voor de data en het verkrijgen van data aan bod komt. Vervolgens begint hoofdstuk 3 aan de uitvoering van het onderzoek door middel van twee pilotexperimenten. Dit leidt tot het wiskundige model dat in hoofdstuk 4 wordt uitgelegd. Iets wat voortvloeit in de testexperimenten van hoofdstuk 5. Dit alles resulteert tot het uiteindelijke onderzoek (d.w.z. vier casestudies) in hoofdstuk 6. Hoofdstuk 7 presenteert het herhalingsexperiment. Dit onderzoek eindigt met een conclusie in hoofdstuk 8 en een discussie in hoofdstuk 9.

Hoofdstuk 1: theoretisch kader

Het theoretisch kader presenteert een literatuurstudie die door middel van vijf bronnen antwoord geeft op de beschouwende deelvragen één en twee. Begrippen staan schuingedrukt en zijn in de begrippenlijst terug te vinden.

Bron 1: grafentheorie en algoritmiek

Inleiding

Grafentheorie is een deelgebied binnen de discrete wiskunde. Daarnaast heeft het toepassing binnen de computerwetenschappen. Bekend is bijvoorbeeld het Travelling Salesman Problem (TSP) dat een onderdeel vormt van de Millenniumprijsproblemen (Clay Mathematics Institute, Z.D.). Daarnaast worden grafen in dit onderzoek gebruikt om snelwegen mee te modelleren. Hierdoor is deze bron een belangrijk onderdeel van het theoretisch kader, omdat het een algemeen, maar diep beeld geeft over grafentheorie.

Deze (online) opleiding, genaamd 'Advanced algorithmics and graph theory with Python,' is door edX in samenwerking met Institut Mines-Télécom (IMT Atlantique) gevormd. Hierdoor kent het veel auteurs met elk andere specialiteiten. De belangrijkste bijdragers zijn:

- Victor Gripon (associate professor bij IMT Atlantique): de co-auteur van circa 70 wetenschappelijke artikelen over onder andere kunstmatige neurale netwerken en signaalverwerking.
- Patrick Meyer (professor bij IMT Atlantique): een expert in operationele wetenschappen, of specifieker 'Multi-Criteria Decision Aiding'.
- Nicolas Farrugia (associate professor bij IMT Atlantique): vooral gespecialiseerd in neurowetenschappen. Hij is ook docent in onder andere deep learning en kunstmatige intelligentie.

Doordat het een brede opleiding is, geeft het een complete uitleg van grafentheorie. Deze compleetheid wordt ondersteund door het feit dat deze bron uit 2022 komt en daarmee de actuele staat van deze wetenschap in kaart brengt. Doordat de bron een opleiding is, is het doel van de universiteit onderwijzen.

Samenvatting

Algemene kenmerken en opbouw van een graaf

Een *graaf* (G) is een verzameling van vertices (V) en edges (E) (zie vergelijking 1). Deze edges vormen een verbinding tussen verschillende vertices (zie vergelijking 2). De grootte van een graaf wordt bepaald door het aantal edges (d.w.z. |E|). De rangschikking is het aantal vertices is (d.w.z. |V|).

$$G = (V, E) \quad (1)$$

$$E = \{\{u, v\} \mid u, v \in V \text{ en } u \neq v\} \quad (2)$$

Hierin zijn *u* en *v* vertices.

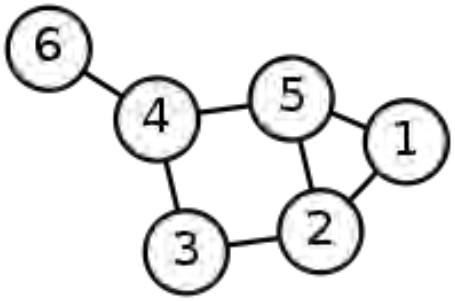
Ook zijn er gewogen grafen. Hierbij wordt aan de edges een bepaald gewicht meegegeven wat bijvoorbeeld een lengte aangeeft. Dit heeft invloed op de werking van verschillende algoritmen. Daarnaast zijn er gerichte grafen. Hier kennen de edges maar één richting en is vergelijking 3 van toepassing.

$$E = \{(u, v) \neq (v, u) \mid u, v \in V \text{ en } u \neq v\} \quad (3)$$

De weergave een graaf gebeurt (naast compleet visueel of abstract) in een *matrixvoorstelling*. Het is matrix waarbij de hoeveelheid rijen en kolommen beide overeenkomen met het aantal vertices. De aanwezigheid van een edge wordt met een 1 aangegeven, terwijl de afwezigheid met een 0 wordt aangegeven. Een voorbeeld hiervan is te zien in figuur 1 en vergelijking 4. Bij gewogen grafen is de 1 vervangen door het gewicht van de edge. In een matrix is met gemak te zien of tussen twee vertices een edge aanwezig is, het is alleen moeilijk om af te lezen welke vertices buren zijn van een bepaalde vertex. Daarom is een matrixvoorstelling in de meeste computerprogramma's vervangen door een dictionary. Deze combineert de voordelen van zowel een lijst als een array (d.w.z. matrix). In een lijst is namelijk eenvoudig te zien welke buren een vertex heeft, terwijl er alleen via een loop (door de gehele lijst) te zien is of er een edge tussen twee vertices aanwezig is.

Figuur 1:

Eenvoudige Graaf bestaande uit 6 Vertices



Noot. Wikipedia, Z.D.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (4)$$

Complexiteit

Computationele algoritmen verwant aan grafentheorie kennen elk een bepaalde *complexiteit*: een maat voor het maximale nummer elementaire operaties die wordt aangegeven als een functie van de input n (hierdoor maakt de grootte van de input niet uit). Binnen dit veld van de computerwetenschappen wordt de Big-O notatie gebruikt. Hierbij vervallen van de functie alle constante en lagere orde termen. Deze notatie wordt gedurende dit gehele onderzoek gebruikt als een maat voor de snelheid en (tijds)complexiteit van een algoritme. Een voorbeeld hiervan is zien in vergelijking 5 en 6.

$$f(n) = 8n^6 + 5n^4 + 7n \quad (5)$$

$$f(n) = O(n^6) \quad (6)$$

Graaf doorkruising: BFS en DFS

Graaf doorkruising richt zich op de toegankelijkheid van alle vertices binnen een graaf. Hierbij wordt een spanning tree gevormd waar geen cykels in voorkomen. Dit gebeurt via een algoritme zoals DFS of BFS. *Depth-First Search (DFS)* doorzoekt alle onbezochte vertices door middel van het *LIFO-principe: last in; first out*. Vanuit een startvertex worden alle buren op een 'stapel' gelegd. Vervolgens wordt steeds de bovenste eraf gepakt en zijn buren worden weer op de stapel gelegd, enzovoorts. Dit leidt er tot dat eerst een gehele tak wordt doorzocht, oftewel Depth-First. Wanneer alle vertices in een tak doorzocht zijn is de vertex die nu nog boven op de stapel ligt een tak of deel van een tak terug. Zo wordt de hele graaf doorzocht. De edges die hierbij doorzocht worden vormen de *spanning tree*. Het voordeel van dit algoritme is dat het een eenvoudig algoritme is. Hierbij wordt alleen niet gegarandeerd de kortste route vanaf de startvertex gevonden. Bij *Breadth-First Search (BFS)* gebeurt dit wel. Dit kent het *FIFO-principe: first in; first out*. Vanuit een startvertex worden alle buren op een 'rij' gezet en vervolgens wordt steeds de voorste van de rij gepakt waarna zijn buren achteraan in de rij moeten aansluiten. Dit leidt tot een algoritme die eerst alle buren van een bepaalde vertex doorzoekt. Vandaar ook de naam Breadth-First. Al is dit algoritme in de praktijk ingewikkelder dan DFS, de kortste route van de startvertex wordt wel gevonden. Een nadeel van beide algoritmen is dat ze niet werken voor gewogen grafen. De tijdscomplexiteit van zowel DFS als BFS is $O(|E|)$.

Dijkstra's algoritme

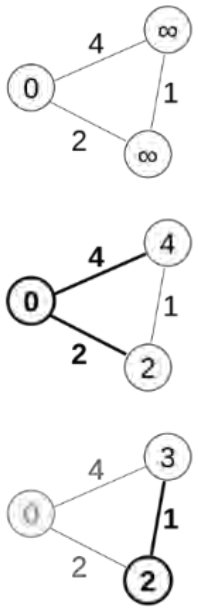
In tegenstelling tot DFS en BFS is *Dijkstra's algoritme* in staat de kortste routes in gewogen grafen te vinden. Het algoritme stamt al uit 1959 en is bedacht door de Nederlandse wiskundige en informaticus Edsger Dijkstra. Dit algoritme is een modificatie op BFS. Vertices worden bezocht op volgorde van toenemend gewicht. Zoals in figuur 2 te zien is, is eerst de afstand van een startvertex naar alle andere vertices als oneindig gedefinieerd. De afstand van de startvertex tot zichzelf wordt als nul gedefinieerd. Vervolgens worden de volgende twee stappen herhaald:

1. Uit een lijst met alle nog onbezochte vertices wordt de buur op de kortste afstand gekozen.
2. Van de gekozen vertex worden de afstanden tot zijn buren bijgewerkt wanneer de nieuwgevonden afstand korter is dan de eerder gevonden afstand.
3. De bezochte vertex wordt verwijderd uit de lijst met onbezochte vertices.

Een voorbeeld hiervan is in figuur 2 geïllustreerd. De kortste afstand (in dit geval 0) wordt gekozen waarna de afstand ∞ (van de beide buren) wordt bijgewerkt naar 4 en 2. Vervolgens wordt de kortste afstand, 2, gekozen, waarna de afstanden bijgewerkt worden. Aangezien $2 + 1 = 3$ en $3 < 4$ wordt de afstand bijgewerkt.

Figuur 2:

Demonstratie van het Dijkstra's Algoritme



Noot. Wikipedia, Z.D.

Dijkstra's algoritme heeft een (tijds)complexiteit van $O(|E| + |V|\ln|V|)$ en is daarmee ingewikkelder dan DFS en BFS. Dit wordt veroorzaakt door het feit dat meerdere keren naar elke buur gekeken moet worden om ervan overtuigd te zijn dat de kortste route gevonden in verband met het gewicht van de edges. Dit is ook in figuur 2 te zien. Terwijl tussen de startvertex tot de bovenste vertex maar één edge zit blijkt via de andere vertex de route korter te zijn.

De vertex op de kortste afstand wordt in een computer algoritme gekozen door middel van een *min-heap*: Een datastructuur waarin data wordt opgeslagen in een (key, value)-pair. Hierin is de key, de vertex en de value, de afstand. In tegenstelling tot het eerdere FIFO- of LIFO-principe kiest men hierbij constant degene met de kleinste afstand, ofwel value, uit de lijst met (key, value)-pairs.

NP-problemen

Grafentheorie kent problemen waarbij (nog) niet de optimale oplossing is gevonden. Het zijn *NP-problemen* dat grofweg hele moeilijke problemen zijn. Hierbij moet men een oplossingen 'gokken' om deze later te verifiëren. Soms zijn deze problemen door middel van een *brute-force algoritme* op te lossen. Deze vinden elke mogelijke oplossing en vergelijken ze allemaal. Dit is alleen mogelijk voor problemen met een kleine input. Een van de bekendste NP-problemen problemen is het Travelling salesmen problem (TSP) waarbij de kortste route door alle vertices in een graaf moet worden gevonden. Wanneer dit probleem door middel van een brute-force algoritme wordt opgelost is de tijdscomplexiteit $O(N!)$. Deze complexiteit is hoog. Zelfs bij 60 vertices moeten hier al $60! = 8{,}32\ldots * 10^{81}$ berekeningen voor gedaan worden. Dit zijn meer berekeningen dan atomen in het universum (Wolfram|Alpha, Z.D.) Deze algoritmen zijn daarom vaak gekoppeld aan een methode genaamd *backtracking* waardoor de tijd wordt ingekort. Indien een gemaakte 'keuze' niet leidt tot de optimale oplossing moet er

volgens backtracking teruggekeerd worden naar het keuzemoment. In grafentheorie, of specifieker TSP, zijn voorbeelden hiervan:

1. Het tijdig stoppen van het brute-force algoritme wanneer de huidige oplossing al slechter is dan de tot nu toe best gevonden oplossing.
2. Vooraf bepalen wat mogelijk de snelste routes zijn door bijvoorbeeld het kiezen van de dichtstbijzijnde vertices.
3. Terug bewegen in een oplossing in plaats van een hele nieuwe oplossing te onderzoeken.

Backtracking heeft geen invloed op de tijdscomplexiteit gegeven dat de Big-O notatie uitgaat van de maximale tijd.

Heuristieken en Greedy algoritmen

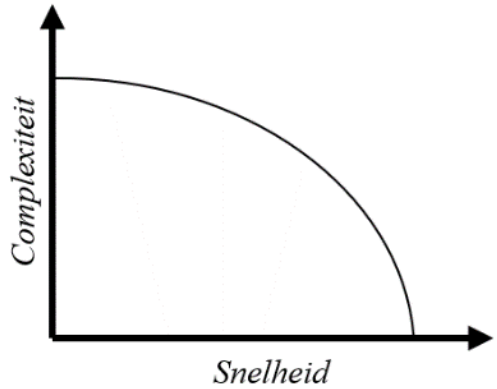
Wanneer inputs bij NP-problemen te groot worden, is het genoodzaakt te werken met *heuristieken algoritmen* die de complexiteit verlagen, maar daarmee een niet-optimale oplossing accepteren. Hierbij maakt men gebruik van intuïtie keuzen die niet ondersteunt zijn. Daarnaast zijn er g*reedy algoritmen* die lokale optima combineren. Een voorbeeld hiervan is bijvoorbeeld het kiezen van de dichtstbijzijnde buur in TSP. Het grote verschil met heuristieken algoritmen is dat bij greedy algoritmen lokale optima mogelijk tot een globale oplossing leiden, terwijl bij heuristieken algoritmen een oplossing gekozen wordt die gegarandeerd een degelijke globale oplossing is.

Complexiteit, correctheid en snelheid

Voor de keuze tussen een brute-force algoritme of een ander algoritme dat een niet-optimale oplossing vindt, moet een keuze tussen snelheid en complexiteit gemaakt worden. Snelheid en complexiteit staan constant met elkaar in tegenstelling. Wanneer men precieze resultaten wil moet men daar lang op wachten. Op basis van de gewenste uitkomst moet er dus een keuze gemaakt worden voor een punt die zich op het Pareto-frontier bevindt dat in figuur 3 zichtbaar is.

Figuur 3:

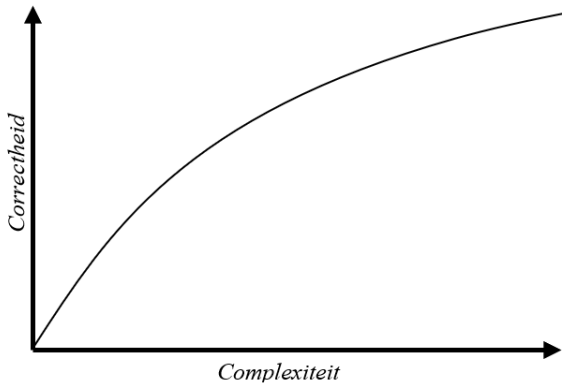
Relatie van de Complexiteit en Snelheid



Daarnaast is complexiteit gerelateerd aan de correctheid. Dit is een manier om verschillende algoritmen met elkaar te vergelijken. Figuur 3 en 4 laten zien dat het soms verstandig is minder correcte algoritme kiezen voor een lagere complexiteit en daarmee een hogere snelheid.

Figuur 4:

Relatie van de Correctheid en Complexiteit.



Bruikbaarheid en relevantie

Door het format van deze bron geeft het een duidelijk beeld over elk belangrijk onderwerp binnen grafentheorie. Het is echter relatief oppervlakkig vergeleken met wetenschappelijke artikel en andere bronnen binnen dit theoretisch kader. Een ander negatief aspect van de bron is dat het weinig praktijkgericht onderwijs met zich meebrengt.

De bron vormt dus een perfect startpunt voor het onderzoek op het gebied van grafentheorie, Daardoor is het een relevant bron voor dit onderzoek. Alle voorgestelde concepten zijn wetenschappelijk onderbouwd en de bron is betrouwbaar. Alle concepten mogen dan ook zonder twijfel in dit onderzoek gebruikt worden.

Verdere verdieping is echter genoodzaakt voor het uitvoeren van dit onderzoek dat complexere concepten bevat. Al gaan de volgende vier bronnen voornamelijk over verkeerstheorie. Grafentheorie komt hier wel terug in de vorm van wiskundige modellen. Daarnaast zijn praktische voorbeelden (met theoretische onderbouwing) te vinden op een platform genaamd CodeCademy. Hiermee is Python geleerd. Enkele van de hierbij horende bronnen zijn in de bronnenlijst opgenomen. De informatie van beide bronnen komt overeen. Doordat de bron is gericht op een ander onderdeel van dit onderzoek dan de andere bronnen binnen dit theoretisch kader is het moeilijk te vergelijken.

Bron 2: verkeersstroommodellering en intelligente verkeerssystemen

Inleiding

Zonder te begrijpen hoe verkeer in elkaar steekt, is men niet in staat verkeer vervolgens filedrukte te verminderen. Daarom is het modelleren van verkeer opgekomen, waardoor verkeer begrepen en verbeterd kan worden. Het is een relatief nieuwe wetenschapstak. Vandaag de dag werpt deze wetenschap elke dag zijn vruchten af op miljoenen weggebruikers in alleen al Nederland. De bron bespreekt de theoretische kant van de verkeerstheorie en vormt de bakermat voor dit onderzoek omdat begrip van deze stof noodzakelijk is voor het maken van modellen en het trekken van conclusies.

Deze (online) opleiding, genaamd 'Intro to traffic flow modeling and intelligent transport systems,' is door edX in samenwerking met École Polytechnique Fédérale de Lausanne (EPFL) gevormd. Het kent veel verschillende auteurs. De belangrijkste hiervan zijn:

- Nicolas Geroliminis (associate professor bij EPFL en hoofd van Urban Transport Systems Laboratory (LUTS)): onder andere assistent-redacteur van 'Transportation Research part C'. Tevens doet hij onderzoek op gebieden van de verkeersstroomtheorie tot grootschalige verkeersnetwerken.
- Anastasios Kouvelas (doctor bij EPFL en directeur van onderzoeksgroep 'Traffic Engineering' bij ETH Zurich): gespecialiseerd in het modelleren, besturen en optimaliseren van grootschalige transportsystemen.
- Raphaël Lamotte (doctor bij EPFL): gespecialiseerd in de werking van file in stedelijke netwerken gericht op keuze voor vertrektijden en nieuwe vormen van mobiliteit.

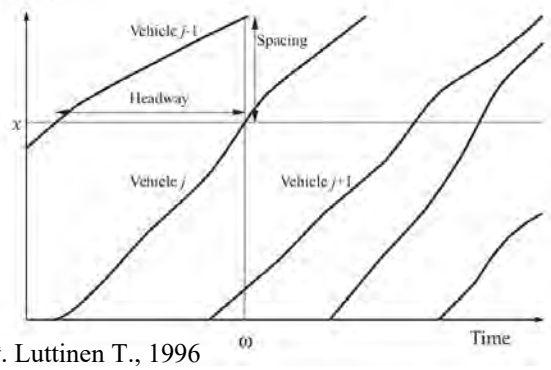
Deze bron is afkomstig uit 2022 en geeft hiermee een accuraat beeld van wetenschap. Tevens is het diepgaand en kent het eenvoudige toegang tot achterliggende wetenschappelijke artikelen, waarvan enkele zijn opgenomen in dit theoretisch kader. Doordat de bron een opleiding is, is het doel van de universiteit onderwijzen.

Samenvatting

Snelheid, intensiteit en dichtheid

Figuur 5:

Voorbeeld van een (Tijd, Afstand)-Diagram



Noot. Luttinen T., 1996

In een (tijd, afstand)-diagram worden de bewegingen van voertuigen weergegeven. Een raaklijn vormt op elk punt de snelheid. Veelvoorkomende begrippen zijn *volgtijd*, wat het tijdsverschil tussen twee voertuigen aangeeft, en *volgafstand*, wat de afstand tussen twee voertuigen aangeeft. Uit dit diagram in figuur 5 zijn de drie belangrijkste begrippen binnen de verkeerstheorie af te leiden. Ten eerste de intensiteit (q in (v(voertuigen)/s)). Hierbij moet het perspectief van een persoon naast de weg worden voorgesteld waarbij de hoeveelheid voertuigen (n) per tijdseenheid (t) geteld wordt (zie vergelijking 7). De totale tijd is ook wel een sommatie van volgtijden (zie vergelijking 9). De intensiteit wordt ook wel berekend door vergelijking 7 en 9 te combineren. Hierbij is $\overline{h}$ de gemiddelde volgtijd is (zie vergelijking 11). Ten tweede is er de dichtheid (K (in v/m)) wat dezelfde principes kent als de intensiteit. Hierbij moet alleen een luchtperspectief worden voorgesteld en wordt er dus over een bepaalde lengte (L) gekeken waarbij er spraken is van de volgafstand.

$$q = \frac{n}{t} \qquad (7) \qquad K = \frac{n}{L} \qquad (8)$$

$$t = \sum_{i=1}^{n} h_i \qquad (9) \qquad L = \sum_{i=1}^{n} S_i \qquad (10)$$

$$q = \frac{n}{\sum_{i=1}^{n} h_i} = \frac{n}{\overline{h}} \qquad (11) \qquad K = \frac{n}{\sum_{i=1}^{n} S_i} = \frac{n}{\overline{S_i}} \qquad (12)$$

Ten derde maakt de snelheid (*v* in (*m/s*)) de relatie tussen de intensiteit en dichtheid compleet (zie vergelijking 13)

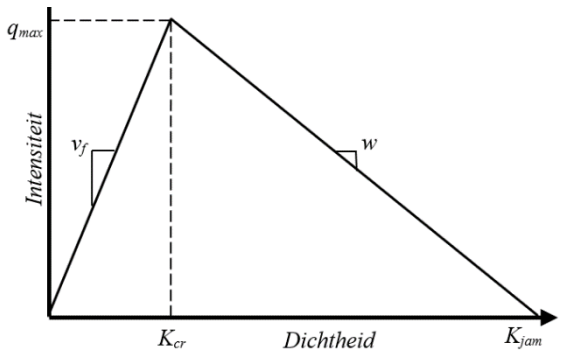
$$v = \frac{q}{K} = \frac{[v/s]}{[v/m]} = [m/s] \qquad (13)$$

In de realiteit is er echter geen spraken van homogene condities, wat er tot leidt dat er een verschil ontstaat tussen het perspectief van de zijkant en het luchtperspectief. Dit leidt tot verschillende snelheden: v_{tms} (tijdsgemiddelde snelheid) afgeleid van de intensiteit en v_{sms} *(*afstandsgemiddelde snelheid) afgeleid van de dichtheid. Hoe dichter de verhouding $\frac{v_{tms}}{v_{sms}}$ bij één ligt, hoe homogener.

Fundamentele diagrammen

Figuur 6:

FD van Intensiteit en Dichtheid met zijn Basisgrootheden



Een *fundamenteel diagram* (*FD*) bestaat uit intensiteit, dichtheid en/of snelheid en geeft inzicht in de verkeersstromen. Deze diagrammen zijn voor veel verkeerssystemen gelijk. De voornaamste is de (K, q)-diagram (zie figuur 6). Officieel is hier spraken van een curve,

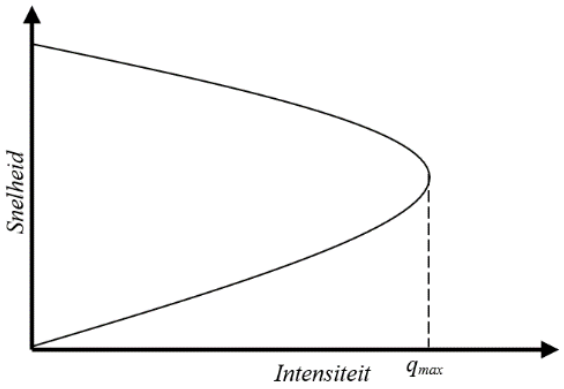
maar deze is te vereenvoudigen tot een driehoek waardoor verschillende begrippen in het diagram zijn af te lezen:

- K_{jam} geeft de maximale dichtheid, het punt waarop de intensiteit nul is terwijl er wel voertuigen aanwezig zijn.
- K_{cr} geeft de kritieke dichtheid aan, het punt waarop de intensiteit gemaximaliseerd wordt.
- q_{max} geeft de maximale intensiteit aan.
- v_f is de vrijestroomsnelheid en duidt op de snelheid waarop er geen congestie is.
- w is de maximale snelheid van een *shockwave* (zie *Hoofdstuk 1; Bron 2; Veranderende verkeerssituaties in FD's*).

Wanneer een intensiteit gegeven is zijn er dus twee mogelijke dichtheden, een zonder congestie (links van de piek) en een met congestie (rechts van de piek). In dit diagram is de paradox van de intensiteit zichtbaar: wanneer er meer vraag is (d.w.z. een hogere dichtheid), wordt er onder condities met congestie juist minder geleverd (d.w.z. intensiteit). Dit is te herkennen aan de dalende lijn (na K_{cr}) in figuur 6.

Figuur 8:

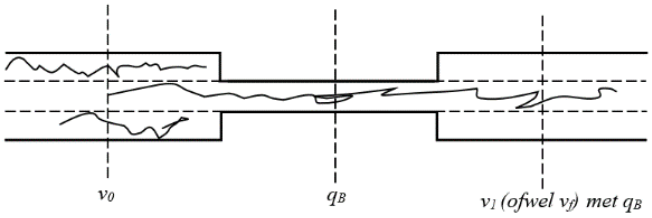
FD van Snelheid en Intensiteit



Veranderende verkeerssituaties in FD's

Figuur 10:

Knelpunt op Snelweg waardoor Intensiteit Afneemt



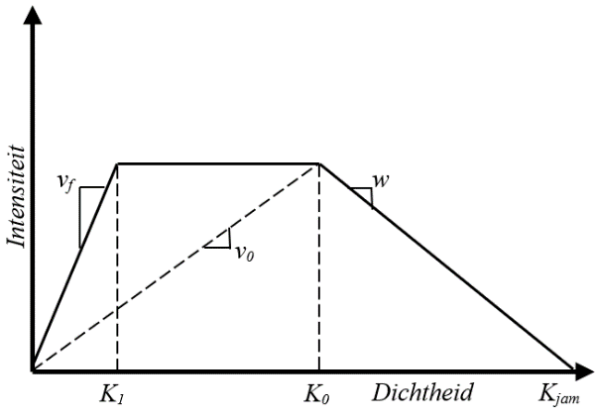
FD nemen door verkeersknelpunten andere vormen aan. Wanneer men bijvoorbeeld te maken heeft met een knelpunt, zoals geïllustreerd in figuur 9, krijgt q een maximale waarde q_B (van bottleneck). Dit heeft als gevolg dat de snelheid (v_0) in een situatie met congestie afneemt en de intensiteit later nooit meer tot zijn maximum kan terugkeren (zie figuur 7) als er laten geen veranderende omstandigheden van toepassing zijn.

Daarnaast zijn er shockwaves plaats: drastische veranderingen in dichtheid die zich voortplanten door een verkeerssysteem. Dit verklaart bijvoorbeeld het harmonica-effect dat men met filerijden tegenkomt. Dit gebeurt alleen voor toenemende dichtheden en deze shockwave verplaatst zich met een bepaalde snelheid (β) door een verkeerssysteem. Deze snelheid is gegeven door de Rankine-Hugoniet conditie (zie vergelijking 14).

$$w = \frac{q_b - q_a}{K_b - K_a} \qquad (14)$$

Figuur 7:

Effecten van Knelpunt op FD

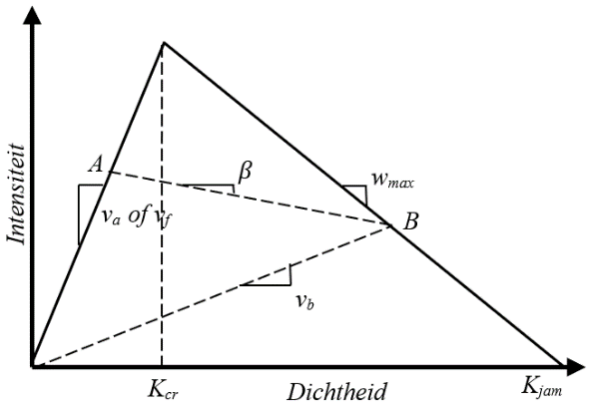


Het FD in figuur 10 laat zien dat de snelheid van een shockwave β gegeven wordt door het verbindingslijn tussen de twee statussen van het verkeer. Deze FD verklaart ook de eerdergenoemde w (of hier, voor de duidelijkheid, w_{max}). Dit is de maximale snelheid van een shockwave die vormt wanneer er een overgang is van K_{cr} naar K_{jam}.

Wanneer er echter een overgang is naar een lagere dichtheid (bij het einde van congestie). Dan spreekt met van de *entropie conditie* waarbij er een pad langs het FD wordt gevolgd tot een maximum voor de intensiteit bereikt is. In dit geval gaat een overgang van *B* naar *A* dus gepaard met het bereiken van q_{max} voor een korte periode.

Figuur 9:

Shockwaves in FD



Een ander fenomeen dat men tegenkomt in empirische data is *capacity drop.* Hierbij volgt een spontane rechte daling net na q_{max} doordat bestuurders chaotische reageren op condities met congestie waardoor de intensiteit afneemt.

Behoudswet

De *behoudswet* stelt dat intensiteit nooit verloren gaat. Voor cellen, die ook in dit onderzoek gebruikt zijn, betekent dit dus dat verlies aan intensiteit gecompenseerd wordt met een verandering in dichtheid. Dit is bewezen in vergelijking 15 t/m 18. Visualisatie hiervan is te zien in figuur 11.

$$n_1 + n_4 = n_2 + n_3 \quad (15)$$

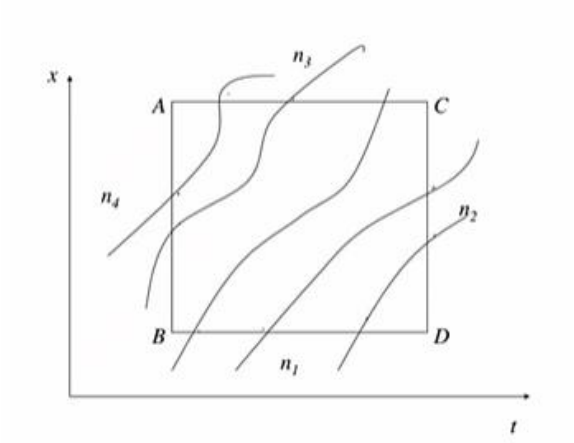
$$q_{BD} * \Delta T + K_{AB} * \Delta L = q_{AC} * \Delta T + K_{CD} * \Delta L \quad (16)$$

$$0 = (q_{AC} - q_{BD}) * \Delta T + (K_{CD} - K_{AB}) * \Delta L \quad (17)$$

$$\frac{\Delta K}{\Delta T} + \frac{\Delta q}{\Delta L} = 0 \quad (18)$$

Figuur 11:

Visualisatie van Behoudswet



Noot. École Polytechnique Fédérale de Lausanne et al., 2022

Microscopisch tegen macroscopisch

Deze samenvatting is gericht op macroscopisch modellen: modellen waar individuele verplaatsingen van voertuigen geen rol speelt. Dit is omdat dit onderzoek alleen gebruikt maakt van macroscopische modellen. Naast macroscopische modellen zijn er microscopische modellen. Deze geven een accurater beeld van de werkelijkheid door hun hogere complexiteit. Ze berekenen namelijk de verplaatsing van elk voertuig apart. Oorspronkelijk werd dit vooral op kleine schaal toegepast (denk bijvoorbeeld aan kruispunten). Tegenwoordig passen de nieuwste wetenschappelijke onderzoeken dit op grote schaal. Dit gaat echter ver buiten het domein van dit onderzoek en wordt daarom niet besproken.

Bruikbaarheid en relevantie

Door het format van deze bron geeft het een begrijpelijk beeld van alle concepten binnen de verkeerstheorie. Daarnaast incorporeert het oefenopdrachten, casestudies ter verduidelijking en verwijzingen naar verdiepende wetenschappelijke artikelen. Dit maakt het op elk gebied een bruikbare bron waarnaar nog vaker gerefereerd wordt. Het is alleen breed en soms niet diepgaand. Dit is ook de reden dat sommige onderwerpen niet besproken zijn en in de overige drie bronnen in meer detail worden besproken aan de hand van wetenschappelijke artikelen. Toch gaf de bron een degelijk beeld over elk onderwerp met duidelijke wiskundige onderbouwing. Deze is om de complexiteit te verminderen grotendeels uit deze samenvatting gelaten.

De bron is de belangrijkste van dit onderzoek door de kennis en de verwijzingen naar wetenschappelijke artikelen. Het is een betrouwbare bron, mede door de oorsprong en publicatiedatum. Hierdoor levert de overmacht van deze bron binnen dit onderzoek geen problemen op. De theorieën zijn wetenschappelijk onderbouwt en hebben daarom geen discussie nodig. In vergelijking tot de andere bronnen binnen dit onderzoek is het de grootste, maar kent het vergeleken met de pure wetenschappelijke artikelen minder diepgang. Elke bevat betrouwbare informatie die tussen alle bronnen overeenkomt. In dit rest van dit literatuuronderzoek wordt daarom specifieker gekeken naar strategieën die in deze opleiding geïntroduceerd zijn.

Bron 3: ramp metering (RM)

inleiding

Ramp metering (RM) is een optimaliseringsstrategie om congestie op snelwegen te verminderen. Aangezien optimaliseringsstrategie in het wiskundig model voorkomen, is deze bron uiterst relevant. Het geeft immers een duidelijk beeld over RM. Daarnaast kent het een interessante casestudie op de Nederlandse A10.

Dit wetenschappelijke artikel, genaamd 'Freeway ramp metering: an overview,' is uitgegeven onder de Technische universiteit van Kreta. De bijdragers zijn:

- Markos Papepgeorgio (doctor bij de Technische universiteit van Kreta): gespecialiseerd in automatische controle- en optimalisatietheorieën en de toepassing hiervan op verkeerssystemen. Hij ontving verschillende prijzen voor zijn werk, zoals de IEEE Trasnportation Award (2022).
- Apostolos Kostialos (anno 2022 werkend voor Hellas S.A., oorspronkelijk onderzoeker en docent bij de Technische universiteit van Kreta): gespecialiseerd in het modelleren en besturen van verkeersstromen en numerieke optimalisatie.

De bron is afkomstig uit 2003, dit vermindert de betrouwbaarheid. Echter, dit is geen probleem omdat RM al lang bestaat en deze bron (in 2022) als verdiepend materiaal werd aanbevolen door de bron besproken in *Hoofdstuk 1; Bron* 2. Het is een wetenschap artikel die als doel heeft een duidelijk beeld te geven over RM. Volgens dit artikel zien we snelwegen nog als een verkeerssysteem zonder verkeerslichten en andere vernuft. Zoals we wegen 100 jaar geleden al kenden.

Samenvatting

Totaal bestede tijd

File verminderen is concreet het verminderen van de totaal bestede tijd (T_s). Deze wordt door middel van vergelijking 19 berekend. Hierin is *N(k)* het aantal voertuigen op tijd *k* een *T* de duur van de tijdstap.

$$T_S = T \sum_{k=1}^{K} N(k) \quad (19)$$

Specifieker bestaat $N(k)$ uit drie termen: de conditie op tijdstip $k = 0$, de instroom (demand (d)) en de uitstroom (supply (s)). Hiervan worden de eerste twee in dit onderzoek als onafhankelijk variabelen gezien en moet de laatste dus geoptimaliseerd worden. Hoe eerder voertuigen immers het netwerk uit stromen, hoe lager de T_s.

Ramp metering

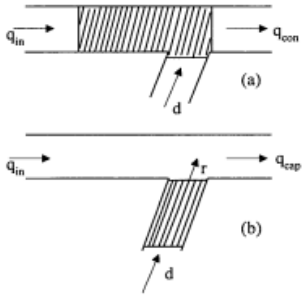
RM is een verkeersstrategie die de input tijdens condities met congestie aanpast zodat de eerder besproken paradox (zie *Hoofdstuk 1; Bron 2*) wordt voorkomen. Zo wordt de intensiteit op maximale capaciteit gehouden. Dit wordt bereikt door verkeer op opritten tegen te houden door middel van stoplichten die per groenlicht maar één voertuig per rijstrook doorlaten. Hierdoor wordt congestie op de snelweg verplaatst naar de oprit (zie figuur 12). Het voordeel voor de T_s is significant en wordt gegeven door het procentuele verschil tussen de T_s^{nc} (no control case) en T_s^{rm} *(*ramp metering):

$$\Delta T_s = \frac{T_s^{nc} - T_s^{rm}}{T_s^{nc}} = \frac{(q_{in}+d-q_{con})-(q_{in}+d-q_{cap})}{(q_{in}+d-q_{con})} \quad (20)$$

$$\Delta T_s = \frac{q_{cap}-q_{con}}{q_{in}+d-q_{con}} \quad (21)$$

Figuur 12:

Visualisatie van RM

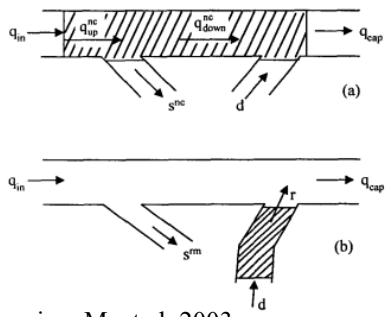


Noot. Papageorgiou, M. et al, 2003

Daarnaast zijn afritten eenvoudiger bereikbaar wat er tot leidt dat de uitstroom verhoogd wordt (zie figuur 13). Naast een verlaging van het T_s, verbetert RM ook de veiligheid door een vermindering van het aantal rijstrookwijzingen en de stress onder bestuurders (waarvan men de negatieve gevolgen ziet in capacity drop). Tevens wordt het invoeggedrag verbeterd en wordt de uitstoot van vervuilende gassen verminderd.

Figuur 13:

Visualisatie van RM



Noot. Papageorgiou, M. et al, 2003

Beperkingen en zijn oplossing

Toch zijn er een aantal beperkingen aan RM:

- Bestuurders die hun individuele reistijd proberen te beperken kiezen mogelijk een andere route.
- Individuele opritten hebben een maximale opslag. Wanneer deze overvol is wordt het andere verkeer tegenhouden. De oprit is dan genoodzaakt om voertuigen door te laten terwijl dit de snelweg niet ten goede is.
- RM op enkele opritten is niet 'eerlijk' voor alle bestuurders omdat andere opritten directe toegang hebben.

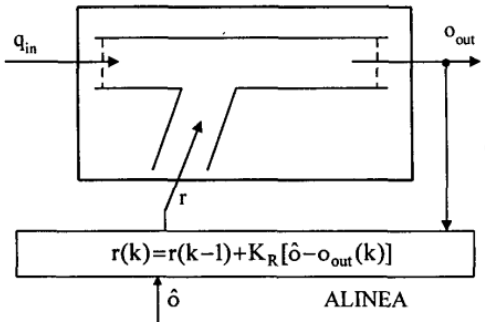
Deze problemen worden opgelost door het toepassen van intelligente RM. Een strategie waarbij verschillende opritten met elkaar samenwerken zodat een tekort aan opslag wordt voorkomen en er eerlijke situaties ontstaan. Hiervoor is een complexer systeem genoodzaakt. Dit gaat buiten het domein van deze studie.

ALINEA

Om de intensiteit van de snelweg constant op de maximale capaciteit te behouden is een intelligent systeem nodig die instroom vanuit de oprit hierop aan past. Een van deze systemen is *ALINEA*.

Figuur 14:

Werking van ALINEA



Noot. Papageorgiou, M. et al, 2003

Hierin staat *o (*occupancy*)* voor de bezetting. Dit is vergelijkbaar met de dichtheid. In het bovenstaande voorbeeld wordt *k* gebruikt als variabele voor tijd. ô is de gewenste bezetting die bijvoorbeeld wordt afgesteld op de kritieke bezetting (vergelijkbaar met K_{cr}). Hierdoor wordt een maximale intensiteit gerealiseerd. Er is hier echter een groot risico op het overslaan naar condities met congestie. Door de bijbehorende capacity drop is het verstandig om de gewenste bezetting (*ô*) lager in te stellen. Vereenvoudigde kennen geen capacity drop, hierdoor is dit niet nodig. K_r is een regulerende parameter waar ALINEA volgens dit onderzoek niet gevoelig voor is. Het voordeel van ALINEA is dat het relatief traag en geleidelijk reageert wat shockwaves voorkomt. Dit is waarom ALINEA de voorkeur krijgt over bijvoorbeeld de vraag-aanbodstrategie die direct, en daarmee te snel, reageert. Dit is een van de andere strategieën die in het artikel aan bod komen.

Casestudie

Dit onderzoek kent een casestudie over de implementatie van RM op de A10. Hierbij wordt historische data gebruikt van uitsluitend de avondspits (d.w.z. 16:00–20:00). Deze data zijn (mogelijk in mindere mate) ook online beschikbaar. Dit maakt deze casestudie samen met gehele artikel uitstekend referentiemateriaal voor dit onderzoek. Voor details van deze casestudie moet het oorspronkelijke onderzoek geraadpleegd worden. Er is een verbetering van 36,6% op de no-control case behaald.

Bruikbaarheid en relevantie

De diepgang van deze bron en de wetenschappelijke onderbouwing maakt het een uiterst bruikbare bron. Geïntroduceerde ideeën worden daarom in de modellen van dit onderzoek gebruikt. Daarnaast biedt deze bron een casestudie en model die als betrouwbaar referentiemateriaal dienen.

Er hoeft niet verder gekeken te worden in de werking van RM aangezien deze bron samen met de vorige bron hier een volledig beeld van geeft. Vergeleken met andere bronnen is deze bron complex en specifiek. Naast de publicatiedatum van deze bron (wat al in de introductie verworpen is), is de bron betrouwbaar door de wetenschappelijke onderbouwing en het feit dat het meer dan 500 wetenschappelijke citaties heeft. Hierdoor hoeft de juistheid niet vergeleken te worden.

Bron 4: variabele snelheidslimieten (VSL)

Inleiding

Terwijl RM een lang bestaande technologie is, zijn *variabele snelheidslimieten* (VSL) een recente technologie waar nog onderzoek naar gedaan wordt. Toch is het een uiterst interessante technologie omdat het mogelijk de totaal bestede tijd (T_s) vermindert en het met andere voordelen gepaard gaat.

Dit wetenschappelijke artikel is uitgegeven onder een project genaamd MSCA IF: ETC-VSL: 'Marie Sklodowska-Curie Actions Individual Fellowships: Efficient Traffic Control with Variable Speed Limits.' Een onderzoeksprogramma opgezet door de Europese Unie die gecoördineerd werd door de Technische Universiteit Delft. De bijdragers aan dit specifieke artikel zijn:

- Markos Papageorgiou (zie *Hoofdstuk 1; Bron 2).*
- Bart de Schutter (professor bij de TU Delft en afdelingshoofd van Delft Centre for Systems and Control (DCSC)): met meer dan 700 publicaties en meer dan 20.000 citaties een van de belangrijkste onderzoekers binnen dit onderzoeksgebied. Bij DCSC geeft hij geeft leiding aan twintig fulltime professors, zestien postdocs en 56 PhD studenten.
- Ionannis Papamichail (professor en directeur bij de Technische universiteit van Kreta): gespecialiseerd in niet-lineaire programmering en optimale controle van snelwegnetwerken.
- José Ramón D. Frejo (post-doctoral onderzoeker bij de Universiteit van Sevilla): gespecialiseerd in voorspellende modellen en zonne-energiesystemen. Hij behaalde onder andere de Outstanding Doctorate Award.

Doordat dit artikel uit 2018 komt, is het verkozen boven andere artikelen. Het onderzoek stelt dat VSL onjuist in verkeersmodellen werd toegepast. Daarom is het doel van dit onderzoek het voorstellen van een nieuw wiskundig model voor VSL.

Voorkennis

Inleiding

Onderzoek binnen de verkeerstheorie volgt elkaar snel op. Dit wordt mede veroorzaakt door de grote belangstelling vanuit bijvoorbeeld overheden. Onderzoek uit 2018 is daarom complex en bouwt voor op eerdere bevindingen. Hierdoor is het genoodzaakt wat benodigde voorkennis te verwerken in de samenvatting. Gelukkig heeft de werking van VSL, dat (in Nederland) gekenmerkt door de borden boven de weg, geen introductie nodig. Het concept spreekt vanzelf.

De voordelen van VSL

VSL kent naast het verminderen van congestie andere voordelen:

- Verbeteren van de veiligheid: VSL verhoogt de homogeniteit onder bestuurders waardoor het gedrag van bestuurders gesynchroniseerd wordt en de hoeveelheid keer dat er gewisseld wordt van rijbaan afneemt (Khondaker, Kattan, 2015).

- Verkeersstoringen oplossen: VSL voorkomt of vertraagd dat verkeer in condities met congestie belandt (Hedgi et al, 2005). Dit gebeurt door een vermindering van de inkomende voertuigstroom.
- Verbeteren van de doorvoer en milieuvoordelen: VSL is in staat de verkeersstroom te verbeteren en daarmee toenemend brandstofgebruik en andere milieunadelen te voorkomen (Zegeye et al., 2010).
- Verminderen van geluidsoverlast.

Beperkingen

VSL kent echter ook zijn beperkingen. Wanneer VSL wordt geïmplementeerd op momenten of plekken waar de verkeerssituatie voor bestuurders nog niet kritiek lijkt, moet er rekening gehouden worden met een lagere naleving dan directere maatregelen zoals RM. Daarnaast leidt VSL tot een hoge dichtheid, wat de volgafstand verminderd (zie *Hoofdstuk 1; Bron 2*). Dit heeft als gevolg dat het samenvoegen van verkeer in secties met meerdere opritten moeilijkheden ervaart. Dit leidt tot congestie op de opritten (Knoop et al, 2010).

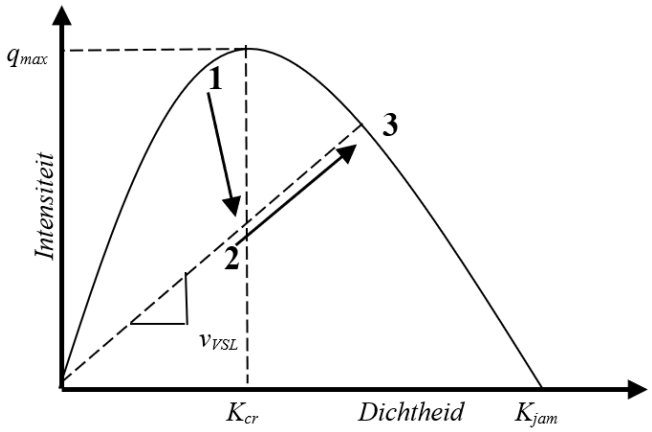
Samenvatting

Voorafgaand aan het model dat de bron zelf introduceert, bespreekt het twee andere bronnen uit de literatuur. Deze vormen een belangrijk startpunt om het voorgestelde model te begrijpen.

Hegyi et al (2005)

Figuur 15:

Effecten van VSL in FD volgens Hegyi et al



Noot. Gebaseerd op Hegyi et al (2005)

Onderzoek uit 2005 stelden dat VSL de intensiteit en dichtheid verandert op een manier die in figuur 15 staat weergegeven. Dit betekent dat wanneer VSL preventief (in situatie 1) wordt ingezet, de situatie in korte tijd naar situatie 2 verplaatst. Door de afname van de snelheid neemt de intensiteit af. Als gevolg hiervan neemt de dichtheid toe. Hierdoor vormt zich een evenwicht tussen situatie 2 en 3.

Voordelen van dit model zijn dat het rekening houdt met de naleving van bestuurders. Echter, het is niet in staat om door de door VSL geïmpliceerde capaciteitsverandering (q_{max}) en kritieke dichtheid (K_{cr}) verandering te bepalen.

$$V_i(k) = \min\left(v_{f,i}\exp\left(\frac{-1}{a_i}\left(\frac{K_i(k)}{K_{c,i}}\right)^{a_i}\right), (1+\alpha_i)V_{c,i}(k)\right) \quad (22)$$

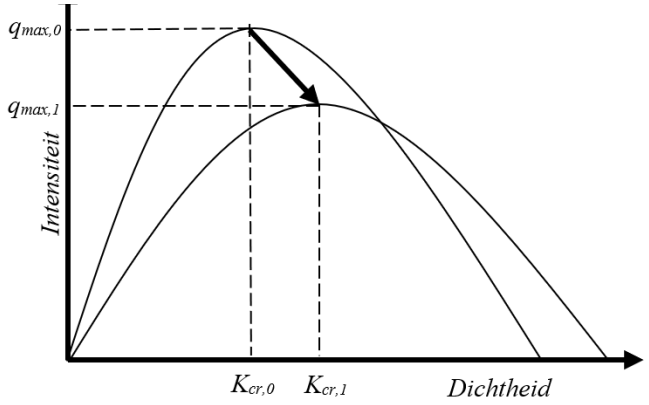
In vergelijking 22 is de formule voor de door VSL geïmpliceerde snelheid te zien. Hierin is $V_i(k)$ de verkregen snelheid, a_i een parameter (die men op empirische data afstemt), $v_{f,i}$ de vrijestroomsnelheid, $K_i(k)$ de dichtheid (die veranderd over tijd), $K_{c,i}$ de kritieke dichtheid, α_i de naleving en $V_{c,i}(k)$ de waarde van de VSL in link i.

Het is van belang te begrijpen dat vergelijking 22 alles bepalend is voor het FD die zien is in figuur 15. Van de oorsprong tot situatie 3 is namelijk de tweede conditie gekozen. Vanaf situatie 3 tot de K_{jam} is de eerste conditie gekozen. Dit volgt uit de relatie $q_i(k) = V_i(k) * K_i(k)$ (zie vergelijking 13).

Clarson et al (2010)

Figuur 16:

Effecten van VSL op FD volgens Clarson et al



Noot. Gebaseerd op Clarson et al, 2010

Het model van Clarson et al (2010) is wel in staat om door VSL verhoogde kritieke dichtheid (K_{cr}) aan te passen. Ditzelfde geldt voor een vermindering van de maximale capaciteit. Het nadeel van dit model is echter dat het geen rekening houdt met de naleving van de bestuurders. Dit model wordt gekenmerkt door een complete verplaatsing van het FD (zie figuur 17).

Vergelijking 23:

Clarson's Model

$$V_i(k) = v^*_{f,i}(k)\exp\left(-\frac{1}{a^*_i(k)}\left(\frac{\rho_i(k)}{\rho^*_{c,i}(k)}\right)^{a^*_i(k)}\right)$$
$$b_i(k) = \frac{V_{c,i}(k)}{V^{max}_{c,i}(k)}$$
$$v^*_{f,i}(k) = v_{f,i}b_i(k)$$
$$\rho^*_{c,i}(k) = \rho_{c,i}\left(1 + A_i(1-b_i(k))\right)$$
$$a^*_i(k) = a_i\left(E_i - (E_i - 1)b_i(k)\right)$$

Noot. Frejo, J. R. D. et al, 2018

De vergelijking voor de snelheid $V_i(k)$ blijft hier onveranderlijk en komt dus overeen met de eerste conditie van vergelijking 22. Enkel wordt hierbij de term $b_i(k)$ toegevoegd. Deze term ligt tussen de 0 en 1 en heeft invloed op alle parameters in de vergelijking zodat de hele FD wordt verschoven, dit is te zien in figuur 16. E_i en A_i zijn parameters die op data worden afgesteld.

Het voorgestelde model

Het model dat wordt voorgesteld in dit onderzoek combineert het model van Hegyi (Hegyi et al, 2005) en het model van Carlson (Carlson et al, 2010). Dit heeft als gevolg dat het model in staat is om zowel K_{cr} te verhogen als q_{max} te verlagen. Daarnaast houdt het model ook rekening met de naleving. Het onderzoek stelt vast dat de naleving voor lage dichtheden klein is.

In het model staat α_i voor de naleving. Er is te zien dat dit model overeenkomt met die van Clarson et al (2010). Enkel, in de $b_i^r(k)$ term is de naleving toegevoegd. Tevens is $v_{f,1}^*(k)$ gewijzigd. Er is immers met $V_{c,i}^{max}$ gerekend (maximale waarde voor de VSL) in plaats van $v_{f,i}$. Volgens het onderzoek is dit realistischer wanneer de maximale VSL groter is dan de vrijestroomsnelheid.

Vergelijking 24:

Voorgestelde Model

$$V_i(k) = v_{f,i}^*(k) \exp\left(-\frac{1}{a_i^*(k)}\left(\frac{\rho_i(k)}{\rho_{c,i}^*(k)}\right)^{a_i^*(k)}\right)$$
$$b_i^r(k) = \min\left(\frac{V_{c,i}(k)}{V_{c,i}^{max}(k)}(1+\alpha_i), 1\right)$$
$$v_{f,i}^*(k) = \min\left(V_{c,i}^{max}(k) b_i^r(k), v_{f,i}\right)$$
$$\rho_{c,i}^*(k) = \rho_{c,i}\left(1 + A_i(1 - b_i^r(k))\right)$$
$$a_i^*(k) = a_i\left(E_i - (E_i - 1) b_i^r(k)\right)$$

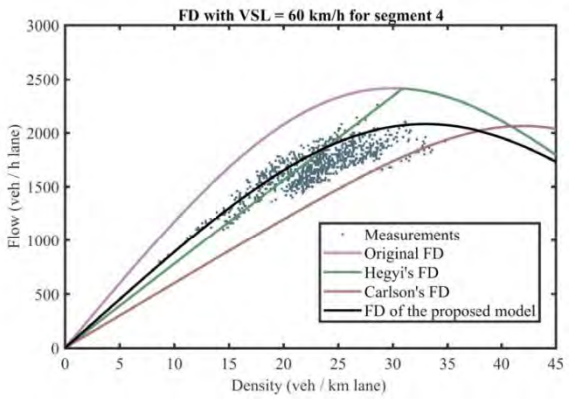
Noot. Frejo, J. R. D. et al, 2018

Het resultaat

Het onderzoek laat zien dat het nieuwe model beter overeenkomt met empirische data (zie figuur 17).

Figuur 17:

Vergelijking van Modellen



Noot. Frejo, J. R. D. et al, 2018

Bruikbaarheid en relevantie

De complexiteit van de bron en het gebrek aan verduidelijking maakte het een bron waarbij naar andere bronnen verwezen moest worden. De bron is specifiek, maar bespreekt niet alle aspecten van VSL. Toch is het een relevante en bruikbare bron omdat de bron verduidelijking schept in de wiskunde achter VSL. Geïntroduceerde concepten worden daarom in het model van dit onderzoek gebruikt. Daarnaast dient de wiskunde als referentiemateriaal.

Naast dat het een andere verkeersstrategie bespreekt, heeft de bron dezelfde vorm als Bron 3 (zie *Hoofdstuk 1; Bron 3*). Door de oorsprong van de bron en publicatiedatum hoeft de inhoud niet met andere bronnen vergeleken te worden, deze zal door de wetenschappelijke onderbouwing juist zijn. Tevens komt het overeen met feiten die in *Hoofdstuk 1; Bron 2* genoemd zijn (École Polytechnique Fédérale de Lausanne, 2022).

Bron 5: cel transmissie model (CTM)

Inleiding

Carlos Daganzo is een van de meest invloedrijke verkeerswetenschappers uit de vorige eeuw. Hij introduceerde het *cel transmissie model (*CTM). Het is hoogst relevant voor dit onderzoek omdat het (na enkele aanpassingen) door middel van grafentheorie geïmplementeerd is om verkeer op snelwegen te modelleren. Hierdoor is gekozen om een samenvatting van de drie invloedrijkste wetenschappelijke artikelen van Daganzo te geven. Zij hebben tezamen duizenden citaties op hun naam staan.

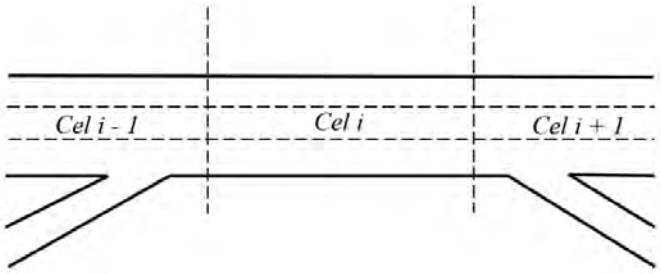
De drie artikelen zijn *'The Cell Transmission Model. Part I: A Simple Dynamic Representation of Highway Traffic'* (1992), *'The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory'* (1994) en *'The cell transmission model, part II: Network traffic'* (1995). Alle zijn uitgegeven onder de Universiteit van Californië, Berkeley en kennen één auteur:

- Carlos Daganzo (Professor in Transportation Engineering bij de Universiteit van Californië): Hij werd in 2019 door de tijdschriftreeks Transportation Research uitgeroepen tot de meest opmerkelijke auteur van de afgelopen 50 jaar met publicaties in vele verschillende onderzoeksrichtingen. Daganzo is nog steeds actief en werkt tegenwoordig in Peking, waar hij hoofdverantwoordelijke is voor het verbeteren van de infrastructuur.

De betrouwbaarheid van deze bronnen moet in twijfel worden genomen door de publicatiedatums. Aan de andere kant vormen deze artikelen de basis voor recentelijk onderzoek. Hierdoor is dus voor deze drie artikelen gekozen omdat ze het duidelijkste beeld over het CTM geven (zonder hier enkel op voort te borduren). In 1992 was het nog niet mogelijk om verkeer op een degelijke manier te modelleren. Daarom is het doel van dit onderzoek het introduceren van een wiskundig model dat in staat is snelwegverkeer te modelleren.

Figuur 18:

Snelweg Opgedeeld in Cellen



Samenvatting

Het discrediteren van tijd en plaats door middel van cellen

Het CTM verdeelt een (deel van een) snelweg of snelwegstructuur in cellen. Deze cellen hebben een afstand van circa ΔL: 100 tot 500 meter. Hierin zijn volgens het model homogene condities. Afstand is dus gediscrediteerd. Ditzelfde geldt voor de tijd, wat in stappen tussen de T: 2 tot 15 seconde is verdeeld.

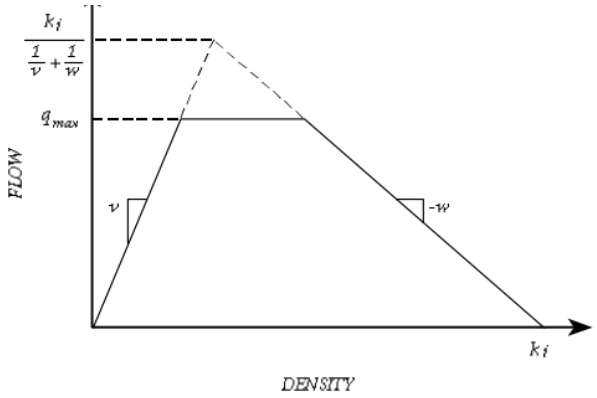
Een belangrijk aspect bij de bepaling van deze afstand en tijd is de *CFL-conditie* (MITx, 2022). Algemeen gezien geldt dat de bij de numerieke bepalingen van differentiaalvergelijkingen, de tijdsstap niet groter mag zijn dan een bepaalde waarde. Anders ontstaan er incorrecte gegevens. In het CTM is dit ook van toepassing. Het is immers niet de bedoeling dat een voertuig in één tijdstap meer dan één cel verplaatst. (Het programma is alleen in staat om voertuigen één cel op te schuiven.) Voor dit specifieke geval van de CFL-conditie geldt vergelijking 25.

$$\mathrm{T} < \frac{\Delta \mathrm{L}}{\mathrm{v_{max}}} \approx \frac{\Delta \mathrm{L}}{\mathrm{v_f}} \qquad (25)$$

Daarnaast is het van belang dat er niet meerdere speciale onderdelen binnen één cel voorkomen. Dit verstoort immers de homogene condities binnen een cel. Denk hierbij bijvoorbeeld aan verschillende op- en/of afritten binnen één cel. Het maximum bestaat uit één oprit en één afrit per cel.

Figuur 19:

FD voor CTM



Noot. Daganzo, 1992

De eigenschappen van een cel

Binnenin een cel zijn spraken van homogene condities die worden weergegeven in een FD. Van tevoren moet men dus onder andere weten wat de capaciteit (q_{max}), maximale dichtheid (K_{jam}) en vrijestroomsnelheid (v_f) is. Daarnaast kent een CTM een aantal speciale cellen die zich voor- en achteraan een reeks (van cellen) bevinden. Beide cellen hebben een oneindige capaciteit. De startcel moet immers voertuigen leveren zonder dat het voertuigen ontvangt. Het ongekeerde geldt voor de laatste cel. Deze ontvangt enkel voertuigen en moet dus in staat zijn alle inkomende voertuigen op te vangen zonder dat dit congestie veroorzaakt.

Verplaatsing door de cellen

Het CTM wordt uitgelegd met cellen die een afstands-tijdsverhouding hebben die gelijk is aan de vrijestroomsnelheid. Hierdoor zijn voertuigen dus in staat om maximaal één gehele cel per tijdsstap te verplaatsen. Dit is om complexiteit in de uitleg en de vergelijkingen te verminderen. Doordat er gerekend wordt met een tijdsstap moet erbij de intensiteit (q) rekening mee worden gehouden dat deze geldt voor de gehele tijdstap (en dus niet voor één seconde). Voor de duidelijkheid is het symbool van q_i naar y_i veranderd ($\mathrm{y}_i = q_i * \Delta T$).

Wanneer het model gestart wordt, vindt verplaatsing van de eerste tot de laatste cel in een reeks plaats. Hiervoor rekent het model de reeks per cel door. Per cel wordt eerst de intensiteit uitgerekend. Hiervoor zijn drie mogelijkheden waaruit het minimum gekozen wordt: het aantal voertuigen in de cel (dat met v_f dus precies één cel verplaatst); de maximale capaciteit; de capaciteit onder condities met congestie. Dit komt overeen met vergelijking 26 waarin de uitstroom voor een cel i naar cel $i + 1$ is berekend.

$$\mathrm{y_i(t)} = \min\{\mathrm{n_i(t)}, \mathrm{Q}_{i+1}, \frac{w_{i+1}}{v_{f,i+1}}\left(N_{i+1} - n_{i+1}(t)\right)\} \qquad (26)$$

Hierin is $n_{i+1}(t)$ het aantal voertuigen in de volgende cel (op een bepaald tijdstip); Q_i het maximale aantal voertuigen dan in een tijdsstap een cel binnen kan stromen en N_i het maximale aantal voertuigen in een cel. Deze vergelijking leidt tot een FD zoals deze te zien is in figuur 19.

De uitstroom van de één is de instroom van de andere. Als ze beide door een cel bepaald zijn, wordt de nieuwe hoeveelheid voertuigen in een cel $(n_i(t+1))$ berekend. Het is belangrijke te beseffen dat dit overeenkomt met de behoudswet (zie vergelijking 18). Hierdoor is deze afleiding gegeven. Dit leidt tot vergelijking 30 en 31.

$$\frac{\Delta \mathrm{K}}{\Delta \mathrm{T}} + \frac{\Delta \mathrm{q}}{\Delta \mathrm{L}} = 0 \qquad (27)$$

$$\Delta \mathrm{K} * \Delta \mathrm{L} = -\Delta \mathrm{q} * \Delta \mathrm{T} \qquad (28)$$

$$\left(\mathrm{K_i(t+1)} - \mathrm{K_i(t)}\right) * \Delta \mathrm{L_i} = -\left(q_i(t) - q_{i-1}(t)\right) * \Delta T \qquad (29)$$

Er geldt $\mathrm{K_i} * \Delta \mathrm{L_i} = \mathrm{n}_i$. Er is immers een dichtheid $(v(voertuigen)/m)$ met een lengte (m) vermenigvuldigd wat een aantal $(voertuigen)$ geeft. Daarnaast wordt er gerekend in tijdstappen. Zoals eerder gesteld is, geldt: $\mathrm{y}_i = q_i * T$. Tevens is $T = \Delta T$.

$$\mathrm{n_i(t+1) - n_i(t) = y_{i-1}(t) - y_i(t)} \quad (30)$$

$$\mathrm{n_i(t+1) = n_i(t) + y_{i-1}(t) - y_i(t)} \quad (31)$$

Vergelijkingen 30 en 31 vormen de basis voor een CTM. Vervolgens worden hier andere concepten aan toegevoegd. Hiervan worden er een aantal besproken.

Invoegen en uitvoegen

Zoals al in figuur 20 te zien is, moet het mogelijk zijn om in- en uit te voegen. Naast de in- en uitstroom van de snelweg zelf kent een cel daarom nog twee stromen: de instroom van de oprit $r_i(t)$ en de uitstroom van de afrit $s_i(t)$.

De instroom van de oprit voegt, onder situaties zonder congestie, samen met de reguliere instroom. Dit leidt tot de vergelijking 32. Dit geldt voor een situatie waarbij er alleen naar invoegen gekeken wordt.

$$\mathrm{n_i(t+1) = n_i(t) + y_{i-1}(t) + r_i(t) - y_i(t)} \quad (32)$$

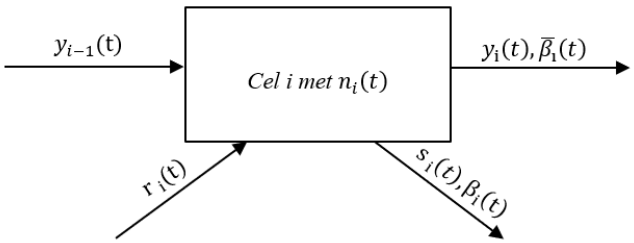
Wanneer een cel de instroom $y_{i-1}(\mathrm{t}) + r_i(t)$ niet aankan (d.w.z. wanneer het maximaal aantal voertuigen N_i gepasseerd wordt), zijn er verschillende mogelijkheden. Ten eerste kan er gekeken worden naar een situatie waarbij al het verkeer op de oprit moet wachten. Dit is te zien in vergelijking 33.

$$r_i = \begin{cases} r_i \; als \; y_{i-1} + r_i \leq R_i \\ R_i - y_{i-1}, \; anders \end{cases} \quad (33)$$

Waarbij $R_i(t)$ de maximaal mogelijke instroom is. De tijdsvariabele is voor de duidelijkheid weggelaten. Dit is echter onwaarschijnlijk. Een deel van de voertuigen op de snelweg en deel van de voertuigen op de oprit houdt namelijk in. Hierdoor is er gerekend met een factor waarmee beide instromen vermenigvuldigd worden om niet boven $R_i(t)$ uit te stijgen.

Figuur 20:

Visualisatie van Intensiteitsstromen in FD



Voor de uitstroom $s_i(t)$ kent men met een *splitsingsverhouding* $\beta_i(t)$. Deze geeft aan welk percentage voertuigen de snelwegen verlaat. Voor de voertuigen die hun weg voortzetten op de snelweg geldt $\overline{\beta_i}(t) = 1 - \beta_i(t)$. De cel kent nu in totaal een grotere maximale uitstroom die omschreven kan worden als $\mathrm{y_i(t) \; / \; \overline{\beta_i}(t)}$.

Wanneer zowel op als afritten in een cel aanwezig zijn, is vergelijking 34 ofwel 35 van kracht.

$$n_i(t+1) = n_i(t) + y_{i-1}(t) + r_i(t) - y_i(t) \; - \; s_i(t) \quad (34)$$

$$\mathrm{n_i(t+1) = n_i(t) + y_{i-1}(t) + r_i(t) - y_i(t) \; / \; \overline{\beta_i}(t)} \quad (35)$$

$$\text{beide met } r_i = \begin{cases} r_i \; als \; y_{i-1} + r_i \leq R_i \\ R_i - y_{i-1}, \; anders \end{cases} \quad (36)$$

Vaststaande routes

Een belangrijke beperking aan het implementeren van afritten, (zoals dat besproken is bij *Hoofdstuk 1; bron 5; Invoegen en uitvoegen*), is dat er geen rekening gehouden wordt met vaststaande routes. Wanneer voertuigen door congestie bijvoorbeeld niet in staat zijn de afrit te nemen, moeten zij in de realiteit in een cel wachten. Nu werkt men echter met een splitsingsverhouding die vaststaande routes niet in acht neemt. Als een afrit gesloten is, nemen alle voertuigen de andere route, omdat ze door de splitsingsverhouding zo verdeeld worden. Ze zijn dus verplicht een andere route te nemen terwijl normaal gesproken congestie zou opbouwen.

Dit valt te verhelpen door het implementeren van een FIFO-principe waardoor een toenemende vertraging (τ) bijhoudt wanneer voertuigen de cel zijn binnengekomen (en het dus als eerst mogen verlaten). Hierbij moet de intensiteit ook gescheiden worden in bestemming om dit juist te verwerken. Dit gaat echter buiten de scope van dit onderzoek en wordt niet geïmplementeerd. De afwijking die nu verkregen wordt moet echter wel beseft worden.

Bruikbaarheid en relevantie

De diepgang en duidelijkheid van de drie bronnen samen maakt het een uiterst bruikbare bron. Het geeft een dieper beeld over een CTM dan *Hoofdstuk 1; Bron 2*. Hierdoor is het CTM ook enkel in deze bron besproken. Dit is van belang om de bijvoorbeeld het implementeren van op- en afritten nu begrepen wordt. Vergeleken met de andere bronnen was het een van de meest specifieke en uitgebreide bronnen die het bruikbaarst is voor het uiteindelijke onderzoek.

Het feit dat deze bronnen heden ten dagen nog geciteerd worden, kenmerkt het feit dat ze zelfs in de snel veranderende wereld van de verkeerstheorie nog relevant zijn. Daarnaast is het te combineren met grafentheorie waardoor het een ondergrond voor het uiteindelijke onderzoek vormt.

Begrippenlijst

De onderstaande bronnenlijst kent de schuingedrukte begrippen uit het gehele theoretisch kader:

Begrip	**Betekenis**
ALINEA	Een intelligent verkeersmanagementsysteem voor RM die de instroom vanaf een oprit aanpast om de maximale capaciteit te behouden.
Backtracking	Wanneer gemaakte 'keuzen' niet leiden tot optimale oplossingen moet men terugkeren naar het keuzemoment.
Behoudswet	Wet die stelt dat intensiteit nooit verloren gaat.
Breadth-First Search (BFS)	Een algoritme dat alle vertices van een graaf door middel van het FIFO-principe doorzoekt. Hierdoor vormt het een spanning tree waarin de kortste routes in een ongewogen graaf te zien zijn.
Brute-force algoritme	Een algoritme dat gegarandeerd de beste oplossing vindt voor een probleem door het gebruik maken van een ongecompliceerde strategie. Hiervoor worden vaak alle mogelijkheden met elkaar vergeleken.
Capacity drop	Spontane rechtlijnige daling in een FD net na q_{max} doordat bestuurder chaotisch reageren op condities met congestie. Hierdoor neemt de intensiteit verder af.
Cel transmissie model (CTM)	Een model voor snelwegen die een snelweg in homogene cellen verdeeld. Voor elke cel worden aparte berekeningen uitvoert.
CFL-conditie	De tijdsstap bij numerieke bepaling van differentiaalvergelijking mag niet groter zijn dan een bepaalde waarde, anders volgen er incorrecte gegevens.
Complexiteit	Een maat voor het maximale aantal elementaire operaties die wordt aangegeven als een functie van de input *n*. Dit wordt vaak aangegeven met behulp van de Big-O notatie.
Depth-First Search (DFS)	Een algoritme dat alle vertices van een graaf door middel van het LIFO-principe doorzoekt. Hierdoor vormt het een spanning tree waarin niet met zekerheid de kortste routes in een ongewogen graaf te zien zijn.
Dijkstra's algoritme	Een algoritme die in staat in de kortste routes in een gewogen graaf te vinden.
Entropie conditie	Een overgang naar een lagere dichtheid waarbij er een pad langs het FD gevolgd wordt tot een maximum voor de intensiteit bereikt is.
FIFO-principe	(First in; First out)-principe. Het object dat als laatste in een 'rij' aansluit, gaat er als laatste weer uit.
Fundamenteel diagram (FD)	Een diagram bestaande uit de intensiteit, de dichtheid en/of de snelheid die inzicht geeft in verkeersstromen.
Graaf	Een verzameling van vertices (V) en edges (E), waarbij edges de verbindingen tussen de vertices vormen.
Greedy algoritme	Een algoritme die lokalen optima combineert.
Heuristieken algoritme	Een algoritme die de complexiteit verlaagt door het maken van intuïtieve keuzen, maar daarmee een niet optimale oplossing accepteert.
LIFO-principe	(Last in; First out)-principe. Het object dat als laatste op een 'stapel' komt, gaat er als eerst weer vanaf.
Matrixvoorstelling	Een matrix die op beide assen het aantal vertices heeft en waarin de aanwezigheid van een edge met een één wordt aangegeven (anders een nul). Bij gewogen grafen wordt de één vervangen door het gewicht van de edge.
Min-heap	Een datastructuur waarin data worden opgeslagen in een zogenaamde (key, value)-pair. Hierbij is de key de vertex en de value de afstand is. Bij het verkrijgen van data wordt degene met de kleinste afstand (ofwel value) als eerst gekozen.
NP-problemen	Hele moeilijke problemen waarbij een oplossing 'gegokt' moet worden om deze later te verifiëren. Er is dus nog geen optimale uitkomst voor gevonden.
Ramp metering (RM)	Een verkeersstrategie die onder congestie de maximale capaciteit wil behouden door het verplaatsen van congestie naar de opritten.
Shockwave	Drastische veranderingen in dichtheid die zich voortplanten door een verkeerssysteem.
Spanning Tree	Een subgraaf van een graaf die alle vertices bevat met een minimaal aantal edges.
Variabele snelheidslimieten (VSL)	Een verkeersstrategie die congestie probeert te verminderen door het impliceren van veranderende snelheidslimieten. Daarnaast brengt het verschillende andere voordelen met zich mee.
Volgafstand	De afstand tussen twee voertuigen.
Volgtijd	Het tijdsverschil tussen twee voertuigen.

Hoofdstuk 2: methodologie

Inleiding

In deze studie is een mixed methods onderzoek uitgevoerd. Dat betekent dat er zowel kwantitatief als kwalitatief onderzoek is uitgevoerd naar de vormingen en het verhelpen van congestie op snelwegen. Hiervoor is na het literatuuronderzoek een wiskundig model geïntroduceerd waarna door middel van deskresearch vier casestudies zijn uitgevoerd. Een overzicht van het gehele proces (inclusief datums) is te lezen in '*Bijlage 1: plan van aanpak (PvA)*'. Voor de validiteit en betrouwbaarheid van dit onderzoek moet de discussie worden geraadpleegd (zie *Hoofdstuk 9*).

Dataverzameling

Voor het literatuuronderzoek is met twee betrouwbare edX opleidingen gestart: één van Institut Mines-Télécom en één van École Polytechnique Fédérale de Lausanne. Deze gingen respectievelijk over grafentheorie en verkeerstheorie. Vanuit hier zijn door middel van de sneeuwbalmethode bronnen gevonden van onder andere de auteurs Daganzo, C. F.; Papageorgiou M.; Schutter, B. D. Deze geven een dieper inzicht in verkeerstheorie. Tevens is Codecademy geraadpleegd om begrip, over de in dit onderzoek gebruikte programmeertaal, Python, te vergroten. Elke fout in de programmeercode heeft immers desastreuze gevolgen. Daarnaast zijn onderzoeken van Rijkswaterstaat en partijen waarmee zij samenwerken geraadpleegd om cases uit dit onderzoek mee te vergelijken.

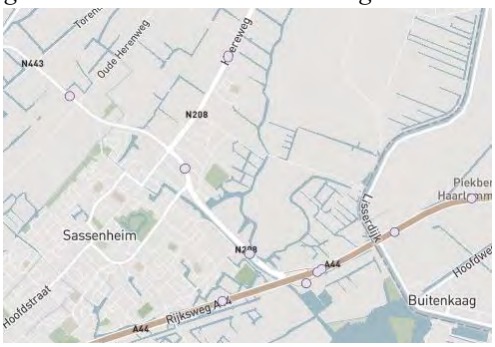
Op basis van de literatuurstudie is een wiskundig model opgesteld. Deze wordt later besproken (zie *Hoofdstuk 4*). Echter, voor dit model is data nodig. Door de omvang van de benodigde data is het niet mogelijk om fieldresearch uitvoeren. De data bestrijken immers meerdere weken en bestaan uit honderden sensoren verspreid over tientallen kilometers snelweg. Daarom kent dit onderzoek deskresearch. Specifieker is het een database-onderzoek waarmee specifieke cases bestudeerd worden. Gelukkig is er een grote hoeveelheid aan open data beschikbaar door nieuwe regelgeving vanuit de Europese Unie, die de beschikbaarheid van open data vergroot (CROW, 2020). Er is dus voor een externe open datasource gekozen. De herkomst, betrouwbaarheid en format van de data wordt in de volgende 3 tussenkoppen besproken.

Herkomst van de data

Verkeersdata worden onder andere verzameld met statische sensoren. Dit zijn bijvoorbeeld detectielussen. Deze bevinden zich onder het wegdek en worden gebruikt voor bijvoorbeeld intelligente verkeerslichten. Ook analyseren ze de verkeerssituatie op snelwegen. Daarnaast zijn er dynamische detectoren. Dit zijn bijvoorbeeld locatiedata van telefoons of taxi's waaruit de snelheden worden afgeleid. Data worden ook verzameld door openbaar vervoer of hulpdiensten. Deze data worden vaak direct verwerkt zodat voorrang wordt verleend aan deze voertuigen. Het voorrang geven van openbaar vervoer zorgt immers voor een vermindering van de totaal bestede tijd (T_s). Dit door de grotere hoeveelheid inzittende (École Polytechnique Fédérale de Lausanne, 2022). Statische en dynamische bronnen verkrijgen andere vormen van data. Deze data moeten gecombineerd worden. Dit is een ingewikkeld proces doordat data mogelijk met andere tijdsstappen verkregen zijn. De meeste vormen van dynamische data zijn echter niet online beschikbaar en dus alleen met eigen onderzoek of in samenwerking met bedrijven of overheidsinstanties verkrijgbaar. Daarom zal dit onderzoek zich richten op statische data.

Figuur 21:

Aanwezigheid van Detectielussen in de regio Sassenheim



Noot. NDW, 2022

Detectielussen

Doordat veel soorten data dus niet verkrijgbaar zijn, maakt dit onderzoek enkel gebruik maken van detectielussen. Dit is in wetenschappelijke onderzoek vaak de gebruikte data-source (École Polytechnique Fédérale de Lausanne, 2022, Papageorgiou, M. et al, 2003). Papageorgiou, M. et al maken bijvoorbeeld gebruik van detectielussen op de (Nederlandse) A10 om onderzoek naar RM te verrichten (zie *Hoofdstuk 1; Bron 3*). Op Nederlandse wegen zijn veel detectielussen aanwezig (zie figuur 21). Dit maakt het een bruikbare en betrouwbare bron van data. Tevens is het online beschikbaar.

Detectielussen maken gebruik van magnetische velden waardoor de tijdsbezetting ($o(t)$ *(occupancy)*) en intensiteit ($q(t)$) gemeten wordt. Vervolgens leidt men de snelheid uit deze twee af. Ten eerste wordt tijdsbezetting omgezet in een schatting voor het aantal voertuigen ($N(t)$) in een bepaald gebied met lengte (ΔL). In dit gebied is van homogene condities spraken (zie vergelijking 37).

$$\mathrm{N(t)} = \Delta\mathrm{L} * \frac{\mu}{\Lambda} * \mathrm{o(t)} \qquad (37)$$

Hierin staat μ voor het aantal rijbanen en Λ voor de gemiddelde lengte van een voertuig.

Volgens berekent men hieruit de totaal bestede tijd (T_s) (zie vergelijking 38) en totaal gereisde afstand (T_d) (zie vergelijking 39) (over bijvoorbeeld één uur). Hieruit volgt een gemiddelde snelheid over een bepaald gebied (zie vergelijking 40).

$$\mathrm{T_s} = \sum_{t=1}^{T} \mathrm{C(t)N(t)} \qquad (38)$$

$$\mathrm{T_d} = \sum_{t=1}^{T} \mathrm{C(t)q(t)L} \qquad (39)$$

$$\mathrm{V_{gem}} = \frac{\mathrm{T_d}}{\mathrm{T_s}} \qquad (40)$$

Hierin staat C(t) (cycle) voor de tijdspan van een meting.

De gekozen databron

Nederland kent verschillende aanbieders van betrouwbare verkeersdata die afkomstig zijn van detectielussen in snel- en provinciale wegen. Deze databronnen zijn gevonden door te starten bij de open data bank van Rijkswaterstaat. Dit omdat het de voornaamste overheidsinstantie is die zich hiermee bezighoudt. Hierdoor is het een betrouwbare bron. Vanuit hier is verder gekeken. De twee voornaamste bronnen zijn vergeleken om de uiteindelijke data-source te bepalen:

- *INWEVA (INtensiteiten op WEgVAkken)*: INWEVA is een databron uitgegeven door Rijkswaterstaat. Dit maakt het een betrouwbare bron. Op circa 4000 wegvlakken wordt de intensiteit gemeten, waarna de overige met een verkeersmodellen worden geschat. Gezien de schattingen door Rijkswaterstaat worden gedaan moet dit geen systematische fouten opleveren. De data zijn alleen niet meer volledig empirisch. Het kent circa tien jaar aan jaargemiddelden van verkeersintensiteit uitgesplitst over dagdelen of uren, voertuigcategorieën en wegdelen.
  Deze bron is vooral geschikt als algemene gegevens van intensiteit genoeg zijn en er vooral onderzoek naar voertuigcategorieën gedaan wordt. Het grote nadeel is dat het geen informatie over de snelheid geeft. Een ander detail is dat het informatie geeft over een baanvak in plaats van een specifieke detector.
- *NDW (Nationaal Dataportaal Wegverkeer)*: bij het DW werkt de Nederlandse overheid samen met particuliere partners aan het inwinnen van verkeersdata. Hierdoor is het een betrouwbare bron. Men selecteert zelf van welke detectielussen er zowel intensiteits- als snelheidsdata wordt ingewonnen. Daarnaast kent het programma veel ingebouwde functies. Hierdoor kunnen periodes en specifieke dagen (zoals alleen werkdagen) geselecteerd worden. Daarnaast is het mogelijk om nationale feestdagen en vakanties uit de data te verwijderen. De data worden gegeven als een uurgemiddelde over de geselecteerde periode. Hierdoor is er meer mogelijk en kent de bron specifiekere data wat de accuraatheid verhoogt.
  Deze bron kent naast de afwezigheid van voertuigcategorieën geen nadelen. Het grote voordeel, ten opzichte van INWEVA, is dat deze bron snelheidsdata kent.
  NDW kent ook een openbare expert module deze brengt nog specifiekere data met zich mee. De grootte van de datasets neemt hierbij echter sterk toe. De gebruikte hardware kan dit niet aan en daardoor wordt hier in dit onderzoek geen gebruik van gemaakt.

Aangezien er geen voertuigdata in dit onderzoek gebruikt worden, gaat de voorkeur uit naar data van in het NDW. Deze bron is tevens betrouwbaarder doordat het geen gebruik maakt van schattingen door middel van modellen.

Dataverwerkingen

Data voor de casestudies zijn op een gestructureerde manier ingewonnen. Ten eerste zijn er onderzoeksgebieden gedefinieerd die binnen de inclusie- en exclusiecriteria van het onderzoeksgebied vallen (zie *Hoofdstuk 2; Inclusie- en exclusiecriteria van het onderzoeksgebied*). Ten tweede zijn voor alle studies datums gekozen die vallen binnen de inclusie- en exclusiecriteria van de periode waarover data gewonnen wordt (zie *Hoofdstuk 2; Inclusie- en exclusiecriteria van de periode*). Vervolgens is er geschetst welke detectielussen nodig zijn per casestudie. NDW levert maximaal de data van twintig detectielussen per download. Hierdoor zijn aangeleverde Excel-bestanden met elkaar samengevoegd om alle data voor één studie te verzamelen. Echter, de format waarin de data geleverd wordt is moeilijk in te lezen door het wiskundige model. Daardoor is de data eerst gestructureerd tot een tabel waarin elke detectielus een kolom vormt in plaats van dat het onder elkaar staat. Nu kan het Excel-bestand worden omgezet in een csv-bestand waarna het wordt ingelezen als een panda's dataframe binnen in het uiterlijke programma. Tevens zijn tijdens het downloaden en structuren van de data aantekeningen gemaakt over welke detectielussen elkaar opvolgen. Dit is gedaan door alle detectielussen met een index aan te geven in oplopende volgorde en op- en afritten apart aan te geven. Verder is aan de data het aantal rijbanen toegevoegd. Dit is direct uit het programma van NDW af te lezen (NDW, 2022). Ten slotte zijn door middel van Google Earth schattingen gemaakt van de lengte van elke cel met een nauwkeurigheid van vijfentwintig meter (Google Earth, Z.D.). Detectielussen worden vaak op gelijke intervallen van elkaar geplaatst waardoor achtereenvolgend dezelfde lengtes genoteerd zijn.

Inclusie- en exclusiecriteria van de periode

Het filteren van de data is van groot belang. Wanneer men de data, die de intensiteit en de snelheid van het verkeer per uur beschrijft, over alle zeven dagen van de week, voor elke dag van het jaar, combineert, komt hier een compleet andere gemiddelde uit dan een waarde die in het echt gemeten wordt. Dit is het geval omdat heterogene gegevens met elkaar gemengd worden waardoor er onbekend gemiddelde ontstaan. In de praktijk brengen werkdagen en weekend, maar ook vakanties en feestdagen, namelijk elk andere gegevens met zich mee. Dit is waarom (degelijke) open-source dataplatformen van verkeersdata ingebouwde functies kennen om data te filteren (NDW, 2022). Er zijn de volgende inclusie- en exclusiecriteria voor de gekozen periode:

- *Werkdagen en weekenden*: werkdagen en weekenden kennen andere gebruiksreden voor snelwegen. Hierdoor wijken bijvoorbeeld avond- en ochtendspitsen af en is er een verschil in het type voertuig dat op de weg aanwezig

Figuur 22:

Autoloze Zondag

Noot. AD, 2019

is. In het weekend zijn er bijvoorbeeld relatief meer personenauto's op de snelwegen aanwezig. Bij het kiezen van een periode moet hier onderscheid tussen worden gemaakt.
- *Feestdagen en (school)vakanties*: feestdagen en vakanties brengen door andere gebruiksredenen andere omstandigheden met zich mee. Dit tast de homogeniteit van de data aan en heeft daarmee en negatief gevolg. Door de lange duur en verspreiding van vakanties moet een afweging gemaakt worden tussen het verlies van de hoeveelheid data of juist de kwaliteit van de data. Ter illustratie hoeven vakanties in het noorden van Nederland mogelijk niet verwijderd te worden bij een onderzoek in het zuiden. Dit is echter afhankelijk van de situatie.
- *Extreme condities*: extreme condities zoals sneeuw of in het verleden autovrije zondagen (al vielen deze in het weekend) moeten uit metingen verwijderd worden omdat deze de gemiddelden verstoren door een onnatuurlijke afwijking. Hierdoor kan er gekozen worden om bijvoorbeeld met de middelste 95% van de metingen te werken. Dit is afhankelijk van de situatie en de gekozen tijdsperiode. In de zomer is het niet nodig, in de winter wel. Corona en die hierbij horende avondklokken en lockdowns zijn ook een voorbeeld van extreme condities.
- *Aanwezigheid van data*: er moet een storingsvrije periode gekozen worden. De afwezigheid van bepaalde detectielussen kan namelijk tot onjuistheden leiden als deze door schatting moet worden opgevuld.

De gekozen periodes verschillen tussen de casestudies en het herhalingsexperiment. Er wordt een periode gekozen waarvoor geldt: alleen werkdagen; buiten feestdagen en vakanties; zonder extreme condities (zomer en geen of weinig coronamaatregelen); periode in 2021 (meest recentelijke periode waarover alle data geheel beschikbaar is). Specifieke keuzen zijn bij de desbetreffende hoofdstukken te lezen (zie *Hoofdstuk 6* en *7*).

Inclusie- en exclusiecriteria van het onderzoeksgebied

De onderzoeksgebieden die voor dit onderzoek gebruikt worden kennen de volgende inclusie- en exclusiecriteria:

- *Detectielussen en data*: er moeten genoeg detectielussen aanwezig zijn zodat een accuraat beeld van de verkeersomstandigheden wordt gevormd. Anders gezegd: de data van het gekozen gebied moet juist en volledig zijn.
- *Congestie*: om congestie te verminderen is het belang dat er zonder maatregelen regelmatig omstandigheden met congestie zijn. Anders kan/hoeft de situatie ook niet verbeterd te worden. Dit geldt dus enkel voor de oranje gebieden in figuur 24. Dit is een visualisatie is van verkeersdata van INWEVA.
- *Zo onafhankelijk mogelijk van andere systemen*: wanneer congestie in een systeem wordt veroorzaakt door congestie in een eerder systeem dat propageert naar het onderzochte systeem, valt eraan het eigenlijke systeem niks te verbeteren, terwijl de condities wel zorgelijk zijn. Dit maakt het uitvoeren van onderzoek onmogelijk en moet daarom voorkomen worden.
- *Mogelijkheid om verkeersstrategieën toe te passen*: gebruikte strategieën, denk aan VSL of RM, moeten mogelijk zijn. Zo moeten er opritten aanwezig zijn voor RM en elektronische verkeersborden voor VSL boven de snelwegen hangen.
- *Vernieuwing*: indien mogelijk moet ervoor een deel van een snelweg of delen van snelwegen gekozen worden die nog weinig bestudeerd zijn om nieuw onderzoek te leveren. Op de Nederlandse snelwegen is dit echter nagenoeg onmogelijk.

Voor de specifieke locaties moeten de desbetreffende hoofdstukken worden geraadpleegd (zie *Hoofdstuk 6* en *7*). Bij elke studie wordt besproken of aan de inclusie- en exclusiecriteria wordt voldaan.

Figuur 23:

INWEVA Data

Noot. Rijkswaterstaat, 2012–2021

Uitvoeren van het onderzoek, ofwel het verwerken van de data

Onderzoek is in deze studie uitgevoerd door middel van een in dit onderzoek voorgesteld wiskundig model. Naast dat deze voor dit onderzoek gebruikt wordt, vormt het ook een van resultaten, ofwel aanbevelingen, van dit onderzoek. Echter, een model moet eerst geprogrammeerd worden. Een minuscule fout is in staat de werking van het hele model teniet te doen. Men probeert daarom problemen van tevoren te verhelpen en te oefenen zodat met zekerheid een juist model gevormd wordt. Hierom zijn verschillende pilotexperimenten uitgevoerd. Ten eerst naar grafentheorie zoals besproken in *Hoofdstuk 1; Bron 1*. Hierdoor zijn problemen in de fundamenten het model voorafgaand aan het uiteindelijke model al verholpen. Tevens zijn de moeilijkheden en de uitdagingen van tevoren genoteerd. Hier wordt bij het programmeren van het uiteindelijke model extra opgelet. Zo is de juistheid van een model gegarandeerd. Dit geldt ook

voor het tweede pilotexperiment dat is uitgevoerd naar een model die een eenvoudig CTM beschrijft. Kennis van de stof is op theoretisch gebied op niveau. Echter, er is weinig praktijk werk verricht. Indien dit niet gedaan wordt, vermindert het de juistheid van het onderzoek.

Voordat het uiteindelijke model geprogrammeerd is, is pseudocode geschreven. Zo is de complexiteit van het model te omvatten. Geschreven code sluit dan mogelijk niet op elkaar aan wat tot fouten leidt. Het herstellen van deze fouten kost meer tijd dan het plannen voorafgaand aan het onderzoek. Na uitvoerige controle en bespreking is de pseudocode omgezet in een computermodel. Vervolgens zijn er eerst vier testexpirimenten uitgevoerd. Casestudies zijn immers complexe verkeerssystemen waarbij elke cel specifieke eigenschappen kent. Fouten in het model zijn dan moeilijk van elkaar te onderscheiden en liggen mogelijk aan de verkeersdata. Hierdoor zijn eerst testexperimenten uitgevoerd. Deze bestaan uit systemen waarin zelfs tussen cellen homogene condities zijn. Hierdoor zijn alle fouten eenvoudig traceerbaar wat later tijd bespaart. Tevens wordt zo de validiteit van het model, en hiermee het onderzoek, verhoogd.

Vervolgens is empirische data geïmporteerd en verwerkt. Er zijn vier casestudies naar filerijke snelwegen in Nederland uitgevoerd. De gekozen perioden en de onderzoeksgebieden vallen binnen de inclusie- en exclusiecriteria. Door het uitvoeren van vier casestudies wordt onderzoekstriangulatie toegepast om de validiteit van de resultaten te behouden. De geldigheid van onderzoeksresultaten wordt immers verhoogd door triangulatie. Door data van casestudies in het model toe te passen is het verkeer gesimuleerd. Vervolgens zijn de resultaten gepresenteerd. Hierbij is door middel van statistisch toetsen gekeken naar waar en hoe file vormt. Tevens zijn problemen in de casestudies verholpen. Vervolgens op handmatige wijze gekeken welke verbeteringen er per casestudie mogelijk zijn door het implementeren van intelligente verkeerssystemen. De keuze voor Nederlandse casestudies is naast nationalisme gebaseerd op het feit dat Nederland uitstekende verkeerssensoren in alle snelwegen bezit. Tevens is er gekozen voor empirische casestudies omdat hierbij gegarandeerd is dat de mogelijke positieve effecten die het model biedt, daadwerkelijk kunnen plaatsvinden. Ten slotte is een herhalingsexperiment uitgevoerd, gedurende een andere periode, om de validiteit van het model en het onderzoek te verhogen.

Concluderen en discussiëren

De resultaten uit verschillende casestudies en testscenario's zijn verwerkt tot een allesomvattende conclusie. Dit is gedaan omdat individuele resultaten vaak niet op elkaar aansluiten. Hierdoor is een overkoepelde conclusie getrokken. Dit is dus het kwalitatieve deel van het onderzoek terwijl de resultaten van specifieke casestudies de kwantitatieve conclusie vormen. Onenigheden tussen verschillende casestudies zijn vervolgens in de discussie besproken. Zo is verzekerd dat het model op juiste wijze handelt en worden mogelijk fouten duidelijk. Tevens is er mogelijk vervolgonderzoek aangeprezen. De verkeerstheorie is namelijk een grote en complex studie. Het kent daarom verschillende andere mogelijkheden. Hier bespreekt men ook de betrouwbaarheid en validiteit van het onderzoek.

Hoofdstuk 3: pilotexperimenten

In deze studie zijn twee pilotexperimenten uitgevoerd; één gericht op grafentheorie en één op verkeerstheorie. Alle code is *Bijlage 4* te vinden.

Pilotexperiment van grafentheorie

Inleiding

In het eerste pilotexperiment zijn alle concepten uit *Hoofdstuk 1; Bron 1* verwerkt tot computermodellen in de programmeertaal Python. Dit dient als verduidelijking en is een geschikte opstap naar het uiteindelijke model. De exacte werking van elke programmeerlijn is achterwegen gelaten. Uitleg over de programmeertaal Python gaat immers buiten de scope van dit onderzoek. De code is wel te zien en er wordt verduidelijking gegeven indien de werking niet identiek is aan of voortbouwt op hetgeen besproken in *Hoofdstuk 1; Bron 1*.

Graaf en vertices

Vergelijking 41:

Definiëren van Graaf en Vertex

```python
# Graph class
class Graph:
    def __init__(self, directed=False):
        self.directed = directed
        self.graph_dict = {}

    def add_vertex(self, vertex):
        self.graph_dict[vertex.value] = vertex

    def add_vertices(self, lst):
        for vertex in lst:
            self.add_vertex(vertex)

    def add_edge(self, from_vertex, to_vertex, weight=0):
        self.graph_dict[from_vertex.value].add_edge(to_vertex.value, weight)
        if self.directed == False:
            self.graph_dict[to_vertex.value].add_edge(from_vertex.value, weight)

    def get_vertex(self, value):
        return self.graph_dict[value]
```

```python
# Vertex class
class Vertex:
    def __init__(self, value):
        self.value = value
        self.edges = {}

    def get_edges(self):
        return list(self.edges.keys())

    def get_edges_weight(self):
        lst_edges = list(self.edges.keys())
        lst_weight = list(self.edges.values())
        lst = []
        for i in range(len(lst_edges)):
            lst.append((lst_edges[i], lst_weight[i]))
        return lst

    def get_edge_weight(self, edge):
        return self.edges[edge]

    def add_edge(self, vertex, weight=0):
        self.edges[vertex] = weight
```

$G = (V, E)$, vergelijking 1 in dit onderzoek, toont dat er twee classes gedefinieerd moeten worden. Een class is een sjabloon voor het creëren van een object. Ten eerste een vertex class die informatie geeft over een specifieke vertex en zijn edges. Ten tweede een graaf class die in staat is specifieke vertices me elkaar te verbinden en tevens aangeeft of de gehele graaf gericht of ongericht is. Deze interactie tussen vertices en de gehele graaf vormt de basis voor een graafstructuur.

DFS en BFS

De werking van DFS en BFS is identiek aan hetgeen besproken in *Hoofdstuk 1; Bron 1*. Belangrijke om op te merken is dat DFS gebruikt maakt van een recursieve functie. Recursie is een computertechniek waar oplossingen van een probleem afhangen van kleinere identieke problemen. In dit geval wordt de functie dus opnieuw opgeroepen wanneer er een nieuwe aangeknoopte vertex is gevonden. Dit voorkomt het gebruik van bijvoorbeeld loops.

Vergelijking 42:

DFS en BFS

```python
# Depth-Frist Search
def dfs(graph, current_vertex, target_value, visited=None):
    if visited is None:
        visited = []
    visited.append(current_vertex)
    if current_vertex == target_value:
        return visited

    for neighbor in graph.graph_dict[current_vertex].get_edges():
        if neighbor not in visited:
            path = dfs(graph, neighbor, target_value, visited)
            if path:
                return path

# Breadth-First Search
def bfs(graph, start_vertex, target_value):
    path = [start_vertex]
    vertex_and_path = [start_vertex, path]
    bfs_queue = [vertex_and_path]
    visited = set()
    while bfs_queue:
        current_vertex, path = bfs_queue.pop(0)
        visited.add(current_vertex)
        for neighbor in graph.graph_dict[current_vertex].get_edges():
            if neighbor not in visited:
                if neighbor == target_value:
                   return path + [neighbor]
                else:
                    bfs_queue.append([neighbor, path + [neighbor]])
```

Dijkstra's algoritme

Vergelijking 43:

Dijkstra's Algoritme

```python
from heapq import heappop, heappush
from math import inf
import random

def dijkstras(graph, start):
    distances = {}
    for vertex in graph.graph_dict.keys():
        distances[vertex] = inf
    distances[start] = 0
    vertices_to_explore = [(0, start)]
    while vertices_to_explore:
        current_distance, current_vertex = heappop(vertices_to_explore)
        for neighbor, edge_weight in graph.graph_dict[current_vertex].get_edges_weight():
            new_distance = current_distance + edge_weight
            if new_distance < distances[neighbor]:
                distances[neighbor] = new_distance
                heappush(vertices_to_explore, (new_distance, neighbor))
    return distances

def visited_all_nodes(visited_vertices):
    for vertex in visited_vertices:
        if visited_vertices[vertex] == 'unvisited':
            return False
    return True
```

Dijkstra's algoritme, de vervanger van BFS wanneer er spraken is van een gewogen graaf, is al uitvoerig besproken. De werking is eenvoudig in de code te herkennen. De oneindige afstand voor de onbezochte vertices en 0 voor startvertex zijn immers eerst gedefinieerd (zie vergelijking 44). Vervolgens is door middel van een min-heap steeds de dichtstbijzijnde vertex gekozen. Vervolgens zijn over de gekozen vertex berekeningen uitgevoerd. Dit is terug te zien in de while-loop (zie vergelijking 43).

Vergelijking 44:

Definiëren Onbezochte Vertices en Startvertex

```python
for vertex in graph.graph_dict.keys():
        distances[vertex] = inf
    distances[start] = 0
```

Travelling salesmen problem (TSP)

Vergelijking 45:

Travelling Salesmen Problem

```
from heapq import heappop, heappush
from math import inf
import random

def visited_all_nodes(visited_vertices):
    for vertex in visited_vertices:
        if visited_vertices[vertex] == 'unvisited':
            return False
    return True

def traveling_salesperson(graph, start_vertex):
    path = ""
    visited_vertices = {x: 'unvisited' for x in graph.graph_dict}
    current_vertex = start_vertex
    visited_vertices[current_vertex] = 'visited'
    path += str(current_vertex) + ' '
    visited_all_vertices = visited_all_nodes(visited_vertices)
    while not visited_all_vertices:
        current_vertex_edges = graph.graph_dict[current_vertex].get_edges()
        current_vertex_edge_weights = {}
        empty_dict = {}
        for edge in current_vertex_edges:
            current_vertex_edge_weights[edge] =
graph.graph_dict[current_vertex].get_edge_weight(edge)
        found_next_vertex = False
        next_vertex = ""
        while found_next_vertex == False:
            if current_vertex_edge_weights == empty_dict:
                break
            next_vertex = min(current_vertex_edge_weights,
key=current_vertex_edge_weights.get)
            if visited_vertices[next_vertex] == 'visited':
                current_vertex_edge_weights.pop(next_vertex)
            else:
                found_next_vertex = True
        if current_vertex_edge_weights == empty_dict:
            visited_all_vertices = True
        else:
            current_vertex = next_vertex
            visited_vertices[current_vertex] = 'visited'
            path += str(current_vertex) + ' '
            visited_all_vertices = visited_all_nodes(visited_vertices)
    print(path)
```

De vorige kleine pilotexperimenten leiden tot een pilotexperiment waar TSP is gecodeerd door middel van een Greedy algoritme. Er is immers steeds voor de dichtstbijzijnde buur gekozen. Het vermindert daarmee de complexiteit in vergelijking tot een brute-force algoritme, enkel neemt de correctheid hiermee wel af (zie *Hoofdstuk 1; Bron 1; Complexiteit, correctheid en snelheid*).

Resultaten en conclusie

Door een onbesproken code die een random graaf construeert van een n aantal vertices zijn de algoritmen getest. Ten illustratie is ervoor een random graaf gekozen met $n = 100$ waarbij elke vertex drie edges aanmaakt. Wanneer DFS en BFS worden toegepast om een route te vinden tussen vertex 10 en vertex 90 geeft dit een resultaat zoals in vergelijking 46.

Vergelijking 46:

Mogelijk Resultaten van respectievelijk DFS en BFS

```
[10, 67, 6, 59, 47, 31, 11, 29, 24, 60, 13, 8, 4,
20, 77, 14, 89, 17, 88, 3, 28, 36, 82, 5, 54, 40,
9, 1, 61, 0, 70, 63, 69, 27, 38, 33, 90]
[10, 11, 86, 90]
```

Voor beide algoritmen wordt een lijst aangemaakt die de route aangeeft. Het verschil tussen DFS en BFS is hier eenvoudig te zien. BFS vindt de optimale oplossing terwijl DFS een lange route vindt, omdat het eerst een hele tak in de diepte doorzoekt. De graaf kent veel edges, dus het doorzoeken van een tak zet lang voort. Er is hierbij een random graaf gevormd waarbij gewicht van edges geen rol speelt. Anders moet er gebruikt worden van het Dijkstra's algoritme. Wanneer elke vertex een gewicht tussen de 1 en 10 krijgt toegeschreven en er met een model van $n = 20$ gewerkt is, leidt dit tot een mogelijk resultaat zoals dat te zien is in vergelijking 47. Hierbij is de startvertex 10. Er is dus een dictionairy gevormd waarin de afstand (value) van een vertex (key) tot de startvertex in aangegeven.

Vergelijking 47:

Mogelijk Resultaat Dijkstra's Algoritme

```
{0:  7, 1:  12, 2:  5, 3:  11, 4:  9,
 5:  10, 6:  13, 7:  9, 8:  10, 9:  7,
 10:  0, 11:  1, 16: 1, 17: 14, 18: 14,
 19:  6}
```

Wanneer TSP wordt toegepast, met als startvertex 10, is een voorbeeld van het resultaat te zien in vergelijking 48. Er is dus te zien dat er vanaf een bepaalde vertex 10 een weg door de gehele graaf gevonden wordt.

Vergelijking 48:

Mogelijk Resultaat TSP

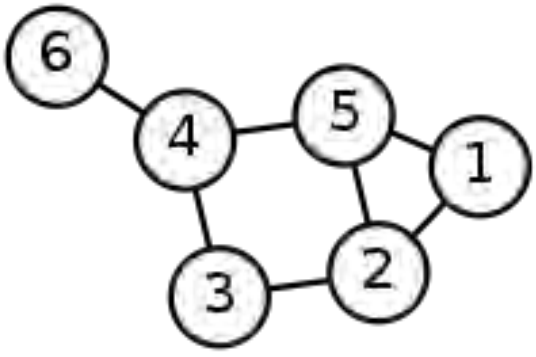
```
10  1 60 92 83 94 50 69 30 87 71 26 29 99 66
21  3 46 75 33 16 24 73 45 74 68 36 91  4 38
5  57 93 14 62 65 35 51 19 64 86 8  82 80 56
61 63 81 17 76 98 77 13 23  2 72 43 47 31 49
27 15 53 70 78 95 9 48 11  67 0  6  59 44 85
58 42 97 18 41 96 89
```

Discussie en bruikbaarheid

De resultaten kennen systematische fouten. Bij TSP moet er een route door alle vertices worden gevonden. In vergelijking 48 zijn echter getallen afwezig. In totaal kent het maar 82 (van de 100) getallen. Een random graaf, die niet is voorbestemd voor TSP, kan zo opgebouwd zijn dat het überhaupt niet mogelijk is om een route te vinden door alle vertices. Ter illustratie is dit uitlegt door middel van figuur 24.

Figuur 24:

Eenvoudige Graaf bestaande uit Zes Vertices



Noot. Wikipedia, Z.D

Wanneer TSP hierop wordt toegepast, waarbij de edges een willekeurige gewichten hebben en de startvertex 2 is, is een mogelijk resultaat te zien in vergelijking 49. Het is niet mogelijk om vanaf 2 een gehele route door alle vertices te vormen aangezien men vastloopt in cel 6. Dit is een probleem waar bij implementatie van TSP rekening mee gehouden moet worden. Toch geven de algoritmen een verhelderd beeld over grafentheorie. Ze zijn bruikbaar en mogen daarom, indien nodig, geïmplementeerd worden in het uiteindelijke model. Verschillende elementen, zoals het opstellen van een graaf en vertex, zullen gegarandeerd terug komen in het model. Mogelijkerwijs is dit onderdeel moeilijk te herkennen in het uiteindelijk model. Echter, het vormt de absolute basis.

Vergelijking 49:

Resultaat TSP van Eenvoudige Graaf uit figuur 2

```
2 3 4 6
```

Pilotexperiment van het CTM

D.m.v. dit pilotexperiment worden het CTM en de theoretische regels van de verkeerstheorie verhelderd. Dit pilotexperiment is daarom compleet gebaseerd op de ideeën gepresenteerd in *Hoofdstuk 1; Bron 5*. Zoals in Vergelijking 50 te zien is, maakt dit model gebruik van grafentheorie om een celstructuur te vormen (zie *Hoofdstuk 3; Pilotexperiment van Grafentheorie; Graaf en vertices*). Er is immers een graaf gemaakt genaamd Highway. Hierin zijn vertices gevormd in de vorm van cellen. Deze zijn later aan elkaar gekoppeld door links, die voor edges staan. Hierdoor ontstaat er een reeks. In de functie update_CTM() vinden vervolgens de basis berekeningen voor een gehele tijdsstap plaats. Deze zijn gebaseerd op vergelijking 26, 30 en 31.

Er is met een aantal bedachten waarden gewerkt en een overschot aan voertuigen in de laatste cellen door daar van tevoren meer voertuigen te plaatsen. Dit leidt tot een shockwave die zich met een bepaalde snelheid achterwaarts door het verkeer heen verplaatst (zie *Hoofdstuk 1; Bron 2; Veranderde verkeerssituaties in FD's*). Figuur 25 laat de resultaten in tien subplots zien waar telkens drie tijdstappen tussen zitten. Hierin is de shockwave duidelijk te herkennen. Door condities met congestie er immers een lagere intensiteit wat voor ophoping zorgt.

Vergelijking 50:

Pythoncode Pilotexperiment van CTM

```
import matplotlib.pyplot as plt

class Highway:
    def __init__(self, directed = True):
        self.directed = True
        self.graph_dict = {}

    def add_cell(self, Cell):
        self.graph_dict[Cell.value] = Cell

    def add_link(self, from_cell, to_cell, weight=0):
        self.graph_dict[from_cell.value].add_link(to_cell.value, weight)
        if self.directed == False:
            self.graph_dict[to_cell.value].add_link(from_cell.value, weight)

    def update_CTM(self):
        lst = []
        for value in self.graph_dict:
            if value == 1 or value == 10:
                pass
            else:
                n = self.graph_dict[value - 1].get_n()
                Q = self.graph_dict[value].get_Q()
                N =
self.graph_dict[value].get_w()/self.graph_dict[value].get_v_f()*(self.graph_dict[value].ge
t_N() - self.graph_dict[value].get_n())
                y_in = min(n, Q, N)
                y_out = min(self.graph_dict[value].get_n(), self.graph_dict[value +
1].get_Q(), self.graph_dict[value + 1].get_w()/self.graph_dict[value +
1].get_v_f()*(self.graph_dict[value + 1].get_N() - self.graph_dict[value + 1].get_n()))
                n_t = self.graph_dict[value].get_n() + y_in - y_out
                self.graph_dict[value].set_n(n_t)
                lst.append(n_t)
        return lst

class Cell:
    def __init__(self, value, K_cr, K_j, Q = 29,  v_f = 30, delta_l = 300, delta_t = 10, n
= 0):
        self.value = value
        self.links = {}
        self.K_cr = K_cr
        self.K_j = K_j
        self.N = K_j * delta_l
        self.Q = Q
        self.v_f = v_f
        self.delta_l = delta_l
        self.delta_t = delta_t
        self.n = n

    def set_n(self, value):
        self.n = value

    def get_n(self):
        return self.n
    def get_v_f(self):
        return self.v_f
    def get_w(self):
        w = (self.v_f * self.K_cr) / (self.K_j - self.K_cr)
        return w
    def get_N(self):
        return self.N
    def get_Q(self):
        return self.Q
    def add_link(self, Cell, weight=0):
        self.links[Cell] = weight

highway = Highway()
B_cell = Cell(1, 0.1, 0.30, n = 10000000000)
cell_2 = Cell(2, 0.1, 0.30)
cell_3 = Cell(3, 0.1, 0.30)
cell_4 = Cell(4, 0.1, 0.30)
cell_5 = Cell(5, 0.1, 0.30)
cell_6 = Cell(6, 0.1, 0.30)
cell_7 = Cell(7, 0.1, 0.30, n = 40)
cell_8 = Cell(8, 0.1, 0.30, n = 40)
cell_9 = Cell(9, 0.1, 0.30, n = 40)
E_cell = Cell(10, 1000, 100000)

highway.add_cell(B_cell)
highway.add_cell(cell_2)
highway.add_cell(cell_3)
highway.add_cell(cell_4)
highway.add_cell(cell_5)
highway.add_cell(cell_6)
highway.add_cell(cell_7)
highway.add_cell(cell_8)
highway.add_cell(cell_9)
highway.add_cell(E_cell)

highway.add_link(B_cell, cell_2)
highway.add_link(cell_2, cell_3)
highway.add_link(cell_3, cell_4)
highway.add_link(cell_4, cell_5)
highway.add_link(cell_5, cell_6)
highway.add_link(cell_6, cell_7)
highway.add_link(cell_7, cell_8)
highway.add_link(cell_8, cell_9)
highway.add_link(cell_9, E_cell)

n_lst = []
for i in range(60):
    lst = highway.update_CTM()
    print(lst)
    n_lst.append(lst)

fig, ax = plt.subplots(nrows=10, ncols=1)
n = 0
for row in ax:
    row.plot(n_lst[n])
    n += 3
plt.xlabel('Cellen')
plt.ylabel('Aantal voertuigen (n_i)')
plt.suptitle('Verplaatsing van een shockwave')
plt.show()
```

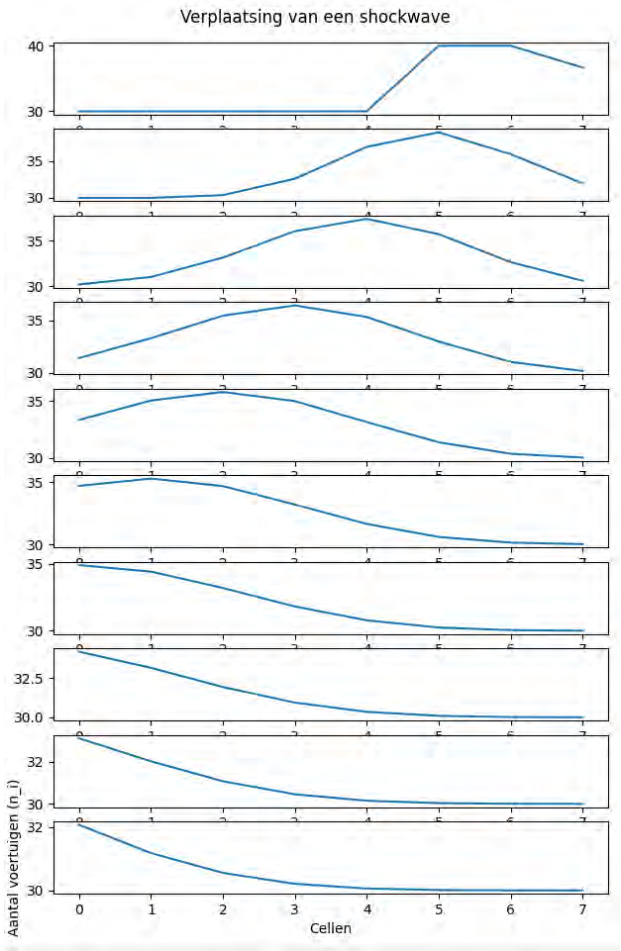
Discussie en bruikbaarheid

Dit model kent een systematische fout. De inputconditie van dit pilotexperiment is incorrect. Om dit te begrijpen wordt een voorbeeld gegeven: stel, voor cel i met $n_i(t)$ voertuigen is berekend dat het y voertuigen naar de cel $i + 1$ verplaatst, dan verandert het aantal voertuigen in cel i direct naar $n_i(t+1) = n_i(t) - y$ Vervolgens gaat het model naar de volgende cel in de lijst met cellen (d.w.z. cel $i + 1$). Voor deze cel wordt de identieke berekening voor de input van cel $i + 1$ uitgevoerd, enkel bezit de cel i nu nog maar $n_i(t+1)$ voertuigen. Dit resulteert in een andere input dan de output en is daarmee in strijd met behoudswet.

Dit probleem is in het uiteindelijke model opgelost door de berekende hoeveelheid y direct als input over te brengen naar cel $i + 1$. Tevens verlaagt dit de hoeveelheid berekeningen die in het hele model moet worden uitgevoerd. Dit pilotexperiment is bruikbaar voor het uiteindelijke model. Het legt namelijk de basis voor het CTM in het uiteindelijke wiskundig model. Daarnaast was het van groot belang problemen, zoals de bovenstaande, van tevoren in te zien. Dit voorkomt systematische fouten in het uiteindelijke model.

Figuur 25:

Visualisatie van Resultaten Pilotexperiment



Hoofdstuk 4: wiskundig model

Inleiding

Het hier voorgestelde wiskundig model vormt het onderzoeksinstrument voor de gehele studie. Het model bestaat uit verschillende delen. Ten eerste bootsten verschillende functie het verkeerverloop na volgens onder andere de theorieën uit *Hoofdstuk 1; Bron 5* beschrijven. Hieraan zijn andere functies verwant die nuances dekken, zoals af- en opritten. Een andere deel van het model probeert de verkeerssituatie te verbeteren door optimaliserende algoritmen. Ten slotte kent het model verschillende functies om de data om te vormen naar de gewenste format. De computercode is aan de rechterkant te zien, terwijl aan de linkerkant uitleg over de code is gegeven. Het gehele model is in *Bijlage 4* te zien. Om de werkingen van het model duidelijker over te brengen wordt het onchronologisch besproken. Verwijzingen naar de code of programmeertermen zijn, ter verduidelijking, schuingedrukt.

Werking van de simulatie

In de basis bestaat het model uit een graaf die bestaat uit cellen. Dit komt overeen met het pilotexperiment (zie *Hoofdstuk 3; pilotexperiment van het CTM*). Er zijn dus twee *classes* die verschillende functies kennen. In wiskunde termen beschrijven functies de relatie tussen één of meer variabelen. In de computerwetenschappen is dit niet anders. Het zijn oproepbare vergelijkingen die de input, een verzameling, omzet in een nieuwe verzameling, de output. In de hier gebruikte programmeertaal Python zijn ze te herkennen aan het woord *def* voorafgaand aan de naam van de functie.

De Cell class

Ten eerste is de *Cell class* gedefinieerd. Dit gebeurt in de *__init__* functie. Voor elke moeten meerdere variabelen worden gedefinieerd. Elke cel kent immers zijn eigen FD (zie *Hoofdstuk 1; Bron 5*). Daarnaast heeft iedere cel variabelen die de aanwezigheid van op- en afritten (*r* en *r_index; s* en *s_index*) en start- en eindcellen (*B_cell* en *E_cell*) aangeven. Andere variabelen worden gebruikt voor de optimaliseringsalgoritmen. Daarnaast wordt een *dictionairy* genaamd *links* bijgehouden die aangeeft welke buren een bepaalde cel heeft.

Vervolgens worden twee waarden uit de startwaarden berekend: N_i, het maximaal aantal voertuigen in een cel, en w_i, de maximale snelheid van een shockwave. Voor de waarden geldt respectievelijk (École Polytechnique Fédérale de Lausanne, 2022):

$$N_i = K_{jam,i} * \Delta L_i * \mu_i \qquad (51)$$

Waarbij ΔL_i de gemiddelde lengte en $\mu_i$het aantal rijbanen is van cel i.

$$w = \frac{q_b - q_a}{K_b - K_a} \quad \text{of specifieker} \quad w_{max,i} = \frac{q_{cr,i} - 0}{K_{cr,i} - K_{j,i}} \qquad (14)$$

Vergelijking 52:

Pythoncode van Wiskundig Model (in delen). Voor de gehele code, zie Bijlage 4

```
class Cell:
    def __init__(self,
                 value,
                 index,
                 lst_delta_l,
                 q_data,
                 lst_Q,
                 lst_v_f,
                 lst_K_cr,
                 lst_K_j,
                 lst_lambda,
                 ramp,
                 r_index,
                 s,
                 s_index,
                 B_cell,
                 E_cell,
                 RM,
                 K_r,
                 VSL,
                 n = 0,
                 a = 10,
                 RM_start = 0,
                 RM_end = 86400,
                 VSL_start = 0,
                 VSL_end = 86400,
                 VSL_start_2 = 86400,
                 VSL_end_2 = 86400,
                 list_cells = 0,
                 v_c = 0,
                 v_c_max = 0,
                 alpha = 0,
                 A = 0,
                 E = 0):
        self.index = index
        self.links = {}
        self.value = value
        self.delta_l = lst_delta_l[index - 1]
        self.q = 0
        self.lst_q = q_data[index]
        self.Q = lst_Q[index - 1]
        self.v_f = lst_v_f[index - 1]
        self.lst_v_f = lst_v_f
        self.K_cr = lst_K_cr[index -1]
        self.lst_K_cr = lst_K_cr
        self.K_j = lst_K_j[index - 1]
        self.lanes = int(lst_lambda[index -1])
        self.n = n
        self.ramp = ramp
        if ramp == True:
            self.lst_r = q_data[r_index] + [0,0,0,0]
            self.input_ratio_onramp = 0
        self.s = s
        if s == True:
            self.lst_s = q_data[s_index]
            self.lst_beta = split_ratio(self)
        self.B_cell = B_cell
        self.E_cell = E_cell
        self.RM = RM
        self.RM_start = RM_start
        self.RM_end = RM_end
        self.K_r = K_r
        self.r = 0
        self.VSL = VSL
        self.VSL_start = VSL_start
        self.VSL_end = VSL_end
        self.VSL_start_2 = VSL_start_2
        self.VSL_end_2 = VSL_end_2
        self.list_cells = list_cells
        self.v_c = v_c
        self.v_c_max = v_c_max
        self.alpha = alpha
        self.A = A
        self.E = E
        self.a = a
        self.N = self.K_j * self.delta_l * self.lanes
        self.w = -(self.v_f * self.K_cr / (self.K_cr - self.K_j))

    def add_link(self, cell, weight=0):
        self.links[cell] = weight

    def get_q(self, t):
        return self.lst_q[t]
    def get_v(self):
        return self.v_f * math.exp(-(1/self.a)*(((self.n / (self.lanes * self.delta_l) /
self.K_cr) ** self.a)))
    def get_r_value(self, t):
        return self.lst_r[t]
    def get_beta(self, t):
        return self.lst_beta[t]
```

Daarnaast moet het model variabelen van cellen aanpassen wanneer een simulatie wordt uitgevoerd. Hiervoor kent de *Cell class* verschillende *set* functies. Bij de eerste negen wordt de waarde enkel aangepast naar een nieuwe. De laatste *set* functie is van belang als een cel een oprit kent. De andere twee *set* functies vormen de interactie tussen de cel en de graaf en zijn dus in staat het aantal voertuigen in de cel aan te passen. De set functies zijn afgeleid van de behoudswet (zie vergelijking 34)

$$\mathrm{n_i(t+1) = n_i(t) + y_{i-1}(t) + r_i(t) - y_i(t) \,/\, \bar{\beta}_i(t)} \quad (34)$$

Deze vergelijking is in dit model opgesplitst in twee delen: de input en de output van een cel. Voor elke cel wordt alleen de output berekend en daardoor gebeurt dit op aparte momenten. Dit is om het eerder aangewezen probleem in *Hoofdstuk 3; Pilotexperiment van het CTM* te voorkomen. Voor de input en output geldt respectievelijk:

$$\mathrm{n_{i+1}(t+1) = n_{i+1}(t) + y_i(t) + r_i(t)} \quad (53)$$

$$\begin{aligned} \mathrm{n_i(t+1)} &= \mathrm{n_i(t) - y_i(t)/\bar{\beta}_i(t)} \\ &= \mathrm{n_i(t) - y_i(t) - s_i(t)} \end{aligned} \quad (54)$$

Extra functies voor de Cell en Highway class en de gekozen parameters

Voor een juiste werking van het model moeten de data zijn omgevormd en moeten de splitsingsverhoudingen van de op- en afritten bekend zijn. Hiervoor kent het model verschillende losstaande functies. De data van het NDW is in uurgemiddelde geleverd (zie *Hoofdstuk 2; De gekozen databron*). Om aan de CFL-conditie te voldoen moet de tijdstap echter klein zijn (zie *Hoofdstuk 1; Bron 5*). De minimale lengte tussen detectielussen bedraagt in dit onderzoek $\Delta L = 250\ (m)$. Wanneer de vrijestroomsnelheid maximaal 120 (km/h) is, geldt voor de maximale tijdstap T:

$$T < \frac{\Delta L}{v_{max}} \approx \frac{\Delta L}{v_f} \approx \frac{250}{(120/3{,}6)} \approx 7{,}50\ (s) \quad (25)$$

Daarnaast is de nauwkeurigheid hoger wanneer er gewerkt wordt met een kleinere tijdstap. Er is gebruik gemaakt van een tijdstap van 5 (s). De data zijn echter aangeleverd in uurgemiddelde. Deze worden daarom omgezet in tijdstappen van 5 seconde. Dit gebeurt in de *adjust_data* functie waarin voor een bepaalde lijst elke waarde in plaats van 1 keer, $\frac{3600}{\Delta T} = \frac{3600}{5} = 720$ keer is toegevoegd. Hierbij worden de eigenlijke data vermenigvuldigd met $\frac{\Delta T}{3600} = \frac{5}{3600}$ om de juistheid van de data te behouden. Vervolgens zijn input- en outputverhoudingen berekend. Deze worden ook wel de splitsingsverhoudingen genoemd. Voor de vier mogelijke splitsingsverhoudingen wordt dit voor elk door middel van een andere functie berekend. Hierbij worden twee waardes opgevraagd waarna de verhouding van één tot de som van de twee wordt bepaald. Dit is van toepassing bij het in- en uitvoegen van voertuigen, bij op- en afritten of bij het splitsen of samenkomen van verschillende snelwegen die beide onderdeel zijn van het onderzoeksgebied.

```
    def set_lst_q(self, lst_q):
        self.lst_q = lst_q
    def set_lst_r(self, lst_r):
        self.lst_r = lst_r
    def set_lst_s(self, lst_s):
        self.lst_s = lst_s
    def set_lst_beta(self, lst_beta):
        self.lst_beta = lst_beta
    def set_input_ratio_onramp(self, input_ratio_onramp):
        self.input_ratio_onramp = input_ratio_onramp
    def set_lanes(self, lanes):
        self.lanes = lanes
    def set_v_f(self, v_f):
        self.v_f = v_f
    def set_K_cr(self, K_cr):
        self.K_cr = K_cr
    def set_q(self, q):
        self.q = q
    def set_a(self, a):
        self.a = a
    def set_input(self, y_out, y_2_out, r_in):
        self.n += y_out + y_2_out + r_in
    def set_output(self, y_out, s):
        self.n -= y_out + s
    def set_input_r(self, r_extra, t):
        self.lst_r[t + 1] += r_extra
    def set_r(self, r):
        self.r = r
```

```
def get_input_ratio(self, cell_1, cell_2):
    input_1 = self.dict[cell_1].lanes * self.dict[cell_1].Q
    input_2 = self.dict[cell_2].lanes * self.dict[cell_2].Q
    return input_1 / (input_1 + input_2)

def split_ratio(self):
    beta_lst = []
    for t in range(len(self.lst_q)):
        beta = self.lst_s[t] / (self.lst_s[t] + self.lst_q[t])
        beta_lst.append(beta)
    return beta_lst

def input_ratio_onramp(self, link, cell):
    input_cell = cell.lanes * cell.Q
    input_r = max(link.lst_r) * 3600 / self.delta_t
    return input_cell / (input_r + input_cell)

def output_ratio(self, lst_links, t):
    return self.dict[lst_links[0]].get_q(t) / (self.dict[lst_links[0]].get_q(t)  +
self.dict[lst_links[1]].get_q(t))

def adjust_data(self):
    timesteps_per_hour =  int(3600 / self.delta_t)
    for cell in self.dict:
        lst_q = []
        for q in self.dict[cell].lst_q:
            for _ in range(timesteps_per_hour):
                lst_q.append(q/timesteps_per_hour)
        self.dict[cell].set_lst_q(lst_q)
        if self.dict[cell].ramp == True:
            lst_r = []
            for q in self.dict[cell].lst_r:
                for _ in range(timesteps_per_hour):
                    lst_r.append(q/timesteps_per_hour)
            self.dict[cell].set_lst_r(lst_r)
        if self.dict[cell].s == True:
            lst_s = []
            lst_beta = []
            for q in self.dict[cell].lst_s:
                for _ in range(timesteps_per_hour):
                    lst_s.append(q/timesteps_per_hour)
            self.dict[cell].set_lst_s(lst_s)
            for beta in self.dict[cell].lst_beta:
                for _ in range(timesteps_per_hour):
                    lst_beta.append(beta)
            self.dict[cell].set_lst_beta(lst_beta)
def get_input_ratio(self, cell_1, cell_2):
    input_1 = self.dict[cell_1].lanes * self.dict[cell_1].Q
    input_2 = self.dict[cell_2].lanes * self.dict[cell_2].Q
    return input_1 / (input_1 + input_2)

def split_ratio(self):
    beta_lst = []
    for t in range(len(self.lst_q)):
```

```
        beta = self.lst_s[t] / (self.lst_s[t] + self.lst_q[t])
        beta_lst.append(beta)
    return beta_lst

def input_ratio_onramp(self, link, cell):
    input_cell = cell.lanes * cell.Q
    input_r = max(link.lst_r) * 3600 / self.delta_t
    return input_cell / (input_r + input_cell)

def output_ratio(self, lst_links, t):
    return self.dict[lst_links[0]].get_q(t) / (self.dict[lst_links[0]].get_q(t)  +
self.dict[lst_links[1]].get_q(t))

def adjust_data(self):
    timesteps_per_hour =  int(3600 / self.delta_t)
    for cell in self.dict:
        lst_q = []
        for q in self.dict[cell].lst_q:
            for _ in range(timesteps_per_hour):
                lst_q.append(q/timesteps_per_hour)
        self.dict[cell].set_lst_q(lst_q)
        if self.dict[cell].ramp == True:
            lst_r = []
            for q in self.dict[cell].lst_r:
                for _ in range(timesteps_per_hour):
                    lst_r.append(q/timesteps_per_hour)
            self.dict[cell].set_lst_r(lst_r)
        if self.dict[cell].s == True:
            lst_s = []
            lst_beta = []
            for q in self.dict[cell].lst_s:
                for _ in range(timesteps_per_hour):
                    lst_s.append(q/timesteps_per_hour)
            self.dict[cell].set_lst_s(lst_s)
            for beta in self.dict[cell].lst_beta:
                for _ in range(timesteps_per_hour):
                    lst_beta.append(beta)
            self.dict[cell].set_lst_beta(lst_beta)
```

De Highway class

De *Highway class* kent de mogelijkheid om cellen toe te voegen aan de *dictionary* die alle cellen in de graaf bijhoudt. Tevens kunnen links tussen cellen worden toegevoegd. Daarnaast kent de class de *run_ctm* functie die verantwoordelijk is voor gehele simulatie. Ten eerst wordt de data in de *adjust-data* functie afgestemd op de gekozen tijdstap. Ten tweede worden de splitsingsverhoudingen voor opritten berekend door een loop door alle cellen, gevolgd door een loop door de ontvangende cellen van de desbetreffende cel (*zie Hoofdstuk 4; Extra functies voor de Cell en Highway class en gekozen parameters*). Vervolgens begint de simulatie die uit een loop bestaat die door alle tijdstappen heenloopt, gevolgd door een loop die door alle cellen heenloopt. Na het resetten van verschillende variabelen, die voor elke tijdstap opnieuw gedefinieerd moeten worden om fouten te voorkomen, bepaalt het model de buren van de onderzochte cel en voegt het deze toe aan een lijst. Vervolgens wordt er gekeken of de cel al bezocht is (en dus in een lijst *lst_cell_2* aanwezig is) of een eindcel is. Dan worden ze overgeslagen. Er nog zijn twee uitzonderingen. Ten eerste kan de buur van een cel een eindcel zijn. Hierbij worden de voertuigen (die in het tijdsinterval de cel kunnen verlaten) verplaatst naar de eindcel. Ook kan een cel een startcel zijn. Voor de output van een normale cel geldt vergelijking 26 (zie *Hoofdstuk 1; Bron 5*).

$$y_i(t) = \min\{n_i(t), Q_{i+1}, \frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)\} \quad (26)$$

Vergelijking 26 geldt voor alle reguliere cellen. Echter, de eerste conditie is niet van toepassing op een startcel. Er worden immers kunstmatig voertuigen aan de cel toegevoegd. $n_i(t)$ wordt daarom vervangen door de intensiteit per tijdstap voorkomend uit een lijst van de empirische data. Ook kan bij de startcel spraken zijn van uitvoegende snelwegen in opvolgende cellen: er moeten dan voertuigen naar twee cellen verplaatst worden. Dit maakt de vergelijking 26 complexer. De output moet nu volgens een berekende splitsingsverhouding verdeeld worden. Voor de output geldt dus dat Q_{i+1} en $\frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)$ voor twee cellen apart berekend moet worden. Het minimum van beide waarde wordt vervolgens bij elkaar opgeteld om zo te vergelijken met $n_i(t)$. Dit is te zien in vergelijking 55 voor cellen $i + 1$ en $i + 2$ zijn.

$$y_i(t) = \min\{n_i(t), min\{Q_{i+1}, \frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)\} + min\{Q_{i+2}, \frac{w}{v_f}\left(N_{i+2} - n_{i+2}(t)\right)\}\} \quad (55)$$

Wanneer ervoor $n_i(t)$ is gekozen worden de voertuigen verdeeld volgens een splitsingsverhouding. Vervolgens wordt er gecheckt of de output de limieten van Q_{i+1} en $\frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)$ niet overschrijdt. Indien dit het geval is wordt het getal vastgesteld om het maximum (van $Q_{i+(1\ of\ 2)}$ of $\frac{w}{v_f}\left(N_{i+(1\ of\ 2)} - n_{i+(1\ of\ 2)}(t)\right)$). Vervolgens wordt het overschot bij de andere cel zijn output geteld. Indien er niet voor $n_i(t)$ gekozen wordt zijn de outputs vastgesteld op de individuele minima van de outputcellen.

```
class Highway: class Highway:

    def __init__(self, delta_t, directed = False):
        self.directed = directed
        self.delta_t = delta_t
        self.dict = {}

    def add_cell(self, cell):
        self.dict[cell.value] = cell

    def add_link(self, from_cell, to_cell, weight=0):
        self.dict[from_cell.value].add_link(to_cell.value, weight)
        if self.directed == False:
            self.dict[to_cell.value].add_link(from_cell.value, weight)

    def run_ctm(self, start, end):
        timesteps = int(86400 / self.delta_t)
        start /= self.delta_t
        end /= self.delta_t
        adjust_data(self)
        TTS = 0
        VSL_check = 0
        VSL_check_2 = 0
        results = {}
        results_v = {}
        n_tot = 0
        for cell in self.dict:
            for _link in self.dict[cell].links:
                        if _link > cell:
                            link = _link
            if self.dict[link].r == True:
                self.dict[link].set_input_ratio_onramp(input_ratio_onramp(self,
self.dict[link], self.dict[cell]))
        for t in range(timesteps):
            if t >= start and t <= end:
                lst_results = {}
                lst_results_v = {}
                lst_cell_2 = []
                for cell in self.dict:
                    lst_links = []
                    r_in = 0
                    _r_in = 0
                    cell_2 = 0
                    y_out_all = 0
                    y_out = 0
                    y_2_out = 0
                    n_1_out = 0
                    n_2_out = 0
                    s = 0
                    s_2 = 0
                    r_in = 0
                    if self.dict[cell].VSL == True:
                        if t > self.dict[cell].VSL_start/self.delta_t and VSL_check == 0:
```

```
                            VSL(highway, self.dict[cell].list_cells, self.dict[cell].v_c,
self.dict[cell].v_c_max, self.dict[cell].alpha, self.dict[cell].A, self.dict[cell].E)
                            VSL_check = 1
                        if t > self.dict[cell].VSL_end/self.delta_t and VSL_check == 1:
                            for cell in self.dict[cell].list_cells:
                                self.dict[cell].set_K_cr(self.dict[cell].lst_K_cr[self.dic
t[cell].index - 1])
                                self.dict[cell].set_v_f(self.dict[cell].lst_v_f[self.dict[
cell].index - 1])
                                self.dict[cell].set_a(10)
                            VSL_check = 2
                        if t > self.dict[cell].VSL_start_2/self.delta_t and VSL_check_2 ==
0:
                            VSL(highway, self.dict[cell].list_cells, self.dict[cell].v_c,
self.dict[cell].v_c_max, self.dict[cell].alpha, self.dict[cell].A, self.dict[cell].E)
                            VSL_check_2 = 1
                        if t > self.dict[cell].VSL_end_2/self.delta_t and VSL_check_2 ==
1:
                            for cell in self.dict[cell].list_cells:
                                self.dict[cell].set_K_cr(self.dict[cell].lst_K_cr[self.dic
t[cell].index - 1])
                                self.dict[cell].set_v_f(self.dict[cell].lst_v_f[self.dict[
cell].index - 1])
                                self.dict[cell].set_a(10)
                            VSL_check_2 = 2
                    for link in self.dict[cell].links:
                        if link > cell:
                            lst_links.append(link)
                    if cell in lst_cell_2:
                        pass
                    elif self.dict[cell].E_cell == True:
                        pass
                    elif len(lst_links) > 0:
                        if self.dict[lst_links[0]].E_cell == True:
                            y_out = self.dict[cell].n * (self.delta_t / 3600 *
self.dict[cell].get_v() / self.dict[cell].delta_l)
                            self.dict[cell].set_output(y_out, s = 0)
                            lst_results[cell] = self.dict[cell].n
                            lst_results_v[cell] = self.dict[cell].get_v()
                        else:
                            if self.dict[cell].B_cell == True:
                                n_1_out = self.dict[cell].get_q(t) * self.dict[cell].lanes
                                Q_N_tot = []
                                for link in lst_links:
                                    Q_out = self.dict[link].Q * self.dict[link].lanes * (1
/ 3600 * self.delta_t)
                                    N_out = self.dict[link].w / self.dict[link].v_f *
(self.dict[link].N - self.dict[link].n)
                                    Q_N_tot.append(min(Q_out, N_out))
                                y_out = min(n_1_out, sum(Q_N_tot))
                                n_tot += y_out

                                if len(lst_links) == 2:
                                    if y_out == n_1_out:
```

```
  output_ratio_highway = output_ratio(self, lst_links, t)
                                        link_1_out = n_1_out * output_ratio_highway
                                        link_2_out = n_1_out * (1 - output_ratio_highway)
                                        if link_1_out > Q_N_tot[0]:
                                            link_2_out += link_1_out - Q_N_tot[0]
                                            link_1_out = Q_N_tot[0]
                                        elif link_2_out > Q_N_tot[1]:
                                            link_1_out += link_2_out - Q_N_tot[1]
                                            link_2_out = Q_N_tot[1]
```

Aangezien de afstand en snelheid, in tegenstelling tot *Hoofdstuk 1; Bron 5*, niet in een verhouding tot elkaar staan die overeenkomt met de tijdstap, moet de hoeveelheid voertuigen die een cel op een tijdstap mag verlaten, aangepast worden op basis van de snelheid in een cel. Deze vergelijking is gebaseerd op vergelijking 22 die geïntroduceerd werd in *Hoofdstuk 1; Bron 4*. Echter, werkt het model in kilometers en uren waardoor ΔT moet worden omgerekend.

$$n_{i,out}(t) = \mathrm{n_i(t)} * \frac{\Delta T}{3600 * \mathrm{v_i(t)}} \ /\Delta L \qquad (56)$$

$$\text{met } \mathrm{v_i\,(t)} \ = \ \mathrm{v_f} \ * \ \exp\left(\frac{-1}{a_i}\left(\frac{K_i(t)}{K_{cr,i}(t)}\right)^{a_i}\right) \qquad (57)$$

Er zijn vier uitzonderlijke situatie die aparte berekeningen vereisen. Om verwarring te vermijden wordt elk apart besproken. Hierdoor komt de volgorde van de code en hetgeen wat besproken wordt niet overeen.

- *Een ontvangende cel met twee inputcellen*: wanneer twee snelwegen bij elkaar invoegen heeft een cel meerdere inputcellen. Ten eerste kijkt het model of dit het geval is door te kijken of de ontvangende cel meer cellen heeft die een kleinere *value* hebben (d.w.z. dat ze aan hem voorafgaan). Indien dit het geval is moet de input netjes verdeeld worden over de twee cellen. Indien dit niet gebeurt zou de reguliere cel al zijn voertuigen verplaatsen naar de volgende cel en andere cel niks omdat de capaciteit als is bereikt door de output van de reguliere cel.
  Dit gebeurt op een manier die vergelijkbaar, maar omgekeerd, is aan een cel met twee ontvangende cellen wat besproken is in *Hoofdstuk 5; Werking van de simulaties; De Highway class*. Hier wordt $\mathrm{n_i(t)}$ namelijk bij elkaar opgeteld. Wanneer het gekozen minimum Q_{i+1} of $\frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)$ bedraagt, worden de outputs volgens een outputverhouding verdeeld. Er wordt gecheckt of dit niet meer is dan $\mathrm{n_i(t)}$. Dit kan het geval zijn indien de voertuigen onevenredig zijn verdeeld. Deze berekeningen zijn in het computerprogramma te herkennen aan *cell_2*.
- *Een cel met twee ontvangende cellen*: dit is al bij de bespreking van de startcel besproken (zie in *Hoofdstuk 5: wiskundig model; Werking van de simulaties; De Highway class)*. Het is in het model te herkennen aan *if len(lst_links) == 2*. Dit betekent dat zich in de lijst met ontvangende cellen twee waarden bevinden. Het heeft immers meerdere ontvangende waarnaar het voertuigen output.

```
                                    else:
                                        link_1_out = Q_N_tot[0]
                                        link_2_out = Q_N_tot[1]
                                if len(lst_links) == 1:
                                    self.dict[lst_links[0]].set_input(y_out, y_2_out=0,
r_in=0)
                                else:
                                    self.dict[lst_links[0]].set_input(link_1_out,
y_2_out=0, r_in=0)
                                    self.dict[lst_links[1]].set_input(link_2_out,
y_2_out=0, r_in=0)
                                lst_results[cell] = self.dict[cell].n
                                lst_results_v[cell] = self.dict[cell].get_v()
                            else:
                                n_1_out = self.dict[cell].n * (self.delta_t / 3600 *
self.dict[cell].get_v() / self.dict[cell].delta_l)
                                n_2_out = 0
                                Q_N_tot = []
                                for _cell_2 in self.dict[lst_links[0]].links:
                                    if _cell_2 < lst_links[0] and _cell_2 != cell:
                                        cell_2 = _cell_2
                                        lst_cell_2.append(cell_2)
                                        n_2_out = self.dict[cell_2].n * (self.delta_t /
3600 * self.dict[cell_2].get_v() / self.dict[cell_2].delta_l)

                                TTS += self.dict[cell].n
                                if cell_2 != 0:
                                    TTS += self.dict[cell_2].n

                                if self.dict[lst_links[0]].ramp == True:
                                    _r_in = self.dict[lst_links[0]].get_r_value(t)
                                    input_ratio_for_onramp = self.dict[lst_links[0]].in-
put_ratio_onramp
                                else:
                                    _r_in = 0

                                n_out = n_1_out + n_2_out + _r_in
                                for link in lst_links:
                                    Q_out = self.dict[link].Q * self.dict[link].lanes * (1
/ 3600 * self.delta_t)
                                    N_out = self.dict[link].w / self.dict[link].v_f *
(self.dict[link].N - self.dict[link].n)
                                    Q_N_tot.append(min(Q_out, N_out))
                                y_out_all = min(n_out, sum(Q_N_tot))
                                n_tot += y_out
                                if len(lst_links) == 2:
                                    if y_out_all == n_out:
                                        output_ratio_highway = output_ratio(self,
lst_links,t)

                                        link_1_out = n_1_out * output_ratio_highway
                                        link_2_out = n_1_out * (1 - output_ratio_highway)
                                        if link_1_out > Q_N_tot[0]:
                                            link_2_out += link_1_out - Q_N_tot[0]
                                            link_1_out = Q_N_tot[0]
                                        elif link_2_out > Q_N_tot[1]:
                                            link_1_out += link_2_out - Q_N_tot[1]
                                            link_2_out = Q_N_tot[1]
                                    else:
                                        link_1_out = Q_N_tot[0]
                                        link_2_out = Q_N_tot[1]

                                if cell_2 != 0:
                                    if self.dict[cell].RM == True and t >
self.dict[cell].RM_start / self.delta_t and t < self.dict[cell].RM_end / self.delta_t:
                                        y_2_out = min(n_2_out, sum(Q_N_tot))
                                        y_out =  max(0, self.dict[lst_links[0]].r +
self.dict[cell].K_r*(self.dict[lst_links[0]].K_cr * self.dict[lst_links[0]].delta_l *
self.dict[lst_links[0]].lanes - self.dict[lst_links[0]].n))
                                        self.dict[lst_links[0]].set_r(y_out)
                                        y_out = min(self.dict[cell].n, y_out)
                                    elif self.dict[cell_2].RM == True and t >
self.dict[cell_2].RM_start / self.delta_t and t < self.dict[cell_2].RM_end /
self.delta_t:
                                        y_out = min(n_out, sum(Q_N_tot))
                                        y_2_out =  max(0, self.dict[lst_links[0]].r +
self.dict[cell_2].K_r*(self.dict[lst_links[0]].K_cr * self.dict[lst_links[0]].delta_l *
self.dict[lst_links[0]].lanes - self.dict[lst_links[0]].n))
                                        self.dict[lst_links[0]].set_r(y_2_out)
                                        y_2_out = min(self.dict[cell_2].n, y_2_out)
                                    else:
                                        if y_out_all != n_out:
                                            input_ratio = get_input_ratio(self, cell,
cell_2)

                                            y_out = y_out_all * input_ratio
                                            y_2_out = y_out_all * (1 - input_ratio)
                                        else:
                                            y_out = n_1_out
                                            y_2_out = n_2_out
                                        if y_out > n_1_out:
                                            y_2_out += y_out - n_1_out
                                            y_out = n_1_out
                                        if y_2_out > n_2_out:
                                            y_out += y_2_out - n_2_out
                                            y_2_out = n_2_out
                                else:
                                    y_out = y_out_all
                                    y_2_out = 0

                                if _r_in != 0:
                                    if self.dict[lst_links[0]].RM == True and t >
self.dict[lst_links[0]].RM_start / self.delta_t and t < self.dict[lst_links[0]].RM_end /
self.delta_t:
                                        y_out = min(n_out, sum(Q_N_tot))
                                        r_in = max(0, self.dict[lst_links[0]].r +
self.dict[lst_links[0]].K_r*(self.dict[lst_links[0]].K_cr *
self.dict[lst_links[0]].delta_l * self.dict[lst_links[0]].lanes -
self.dict[lst_links[0]].n))
                                        self.dict[lst_links[0]].set_r(r_in)
```

- *Een ontvangende cel met een oprit*: wanneer de ontvangende cel ook een oprit kent, vraagt het model ten eerste de gegevens van de oprit op. Dit is de intensiteit op een bepaald tijdstip. Vervolgens wordt deze waarde meegerekend in de totale input van de ontvangende cel $n_i(t)$. Hiermee worden de standaard berekening uitgevoerd. Achter wordt er gekeken of ervoor Q_{i+1} of $\frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)$ gekozen is. Indien dit het geval is, wordt met een inputverhouding de totale input vanuit de oprit en vanuit de reguliere snelweg bepaald. Het overschot op de oprit, ofwel het deel dat (nog) niet de snelweg op kan, wordt verplaatst naar de volgende tijdstap waar opnieuw gekeken wordt of de voertuigen kunnen invoegen op de snelweg. Wanneer ervoor $n_i(t)$ is gekozen rijden alle voertuigen vanaf de oprit de snelweg op.
  Daarnaast wordt er gecheckt of de door de inputverhouding geïmpliceerde waarden niet groter zijn dan de hoeveelheid voertuigen in de cel. Dit gebeurt enkel onder uitzonderlijke omstandigheden, maar er mag geen negatief aantal voertuigen overblijven.
  In het model zijn deze berekeningen te herkennen aan alle berekeningen die *r* kennen.
- *Een cel met een afrit*: Ten slotte kan een cel een afrit hebben. De berekeningen hierbij zijn eenvoudig en zijn niet verdeeld over het gehele model. Daarom zijn onderaan de code zichtbaar.
  Nadat er gekeken is of de cel überhaupt een afrit heeft, wordt er gekeken of ervoor $n_i(t)$ is gekozen. Indien dit het geval is worden $s(t)$ en de nieuwe $y_i(t)$ volgens respectievelijk, de al in de cel bepaalde splitsingsverhouding, $\beta_i(t)$ en $\overline{\beta}_i(t)$ verdeeld. (zie *Hoofdstuk 1; Bron 5*). Wanneer ervoor één van de andere twee mogelijkheden is gekozen (d.w.z. Q_{i+1} of $\frac{w}{v_f}\left(N_{i+1} - n_{i+1}(t)\right)$), benut men de output maximaal en wordt gekeken of het mogelijk is om naast $s(t)$ niet de aangepaste $y_i(t)$, maar de eerder bepaalde $y_i(t)$ te verplaatsen. Deze cel kan dit aan als de output niet gemaximaliseerd wordt door de hoeveelheid voertuigen in de outputcel. Er wordt gecheckt of er door de splitsingsverhouding niet meer dan de oorspronkelijke $n_i(t)$ worden verplaatst naar de volgende cel. Dit kan het geval zijn in uitzonderlijke omstandigheden. Indien dit het geval is, wordt beide waarden met een verhouding verminderd van $\frac{n_i(t)}{s(t)+y_i(t)}$.

Ten slotte worden de voertuigen verplaatst. Dit gebeurt door middel van de *set* functies die al besproken zijn (zie *Hoofdstuk 5; Werking van de simulaties; De cell class*). Deze stappen worden vervolgens voor alle tijdstappen uitgevoerd. Alle resultaten worden aan *dictionaries* toegevoegd zodat later bijvoorbeeld de T_s (totale bestede tijd) berekend kan worden.

```
                                    if r_in < _r_in:
                                        self.dict[lst_links[0]].set_input_r(_r_in -
r_in, t)
                                    r_in = min(r_in, _r_in)
                                else:
                                    if y_out_all != n_out:
                                        r_jam = _r_in - y_out_all * (1 - input_ra-
tio_for_onramp)
                                        self.dict[lst_links[0]].set_input_r(r_jam, t)
                                        r_in = y_out_all * (1- input_ratio_for_onramp)
                                        y_out = y_out_all * input_ratio_for_onramp
                                    else:
                                        r_in = _r_in
                                    if y_out > n_1_out:
                                        r_in += y_out - n_1_out
                                        y_out = n_1_out
                                    if r_in > _r_in:
                                        y_out += r_in - _r_in
                                        r_in = _r_in

                            if self.dict[cell].s == True:
                                if y_out == n_out:
                                    s = y_out * self.dict[cell].get_beta(t)
                                    y_out *= (1 - self.dict[cell].get_beta(t))
                                else:
                                    s = y_out * self.dict[cell].get_beta(t)
                                    if s + y_out > n_out:
                                        ratio = n_out / (s + y_out)
                                        s *= ratio
                                        y_out *= ratio
                            else:
                                s = 0
                            if cell_2 != 0:
                                if self.dict[cell_2].s == True:
                                    if y_2_out == n_2_out:
                                        s_2 = y_out * self.dict[cell_2].get_beta(t)
                                        y_2_out *= (1 - self.dict[cell_2].get_beta(t))
                                    else:
                                        s_2 = y_2_out * self.dict[cell_2].get_beta(t)
                                        if s_2 + y_2_out > n_2_out:
                                            ratio = n_2_out / (s_2 + y_2_out)
                                            s_2 *= ratio
                                            y_2_out *= ratio
                                else:
                                    s_2 = 0
                                self.dict[cell].set_output(y_out, s)
                                if cell_2 != 0:
                                    self.dict[cell_2].set_output(y_2_out, s_2)
                                if len(lst_links) == 1:
                                    self.dict[lst_links[0]].set_input(y_out, y_2_out,
r_in)
                                    self.dict[lst_links[0]].set_q(y_out + y_2_out + r_in)
                                else:
```

```
                                    self.dict[lst_links[0]].set_input(link_1_out, y_2_out,
r_in)
                                    self.dict[lst_links[0]].set_q(link_1_out + y_2_out +
r_in)
                                    self.dict[lst_links[1]].set_input(link_2_out, y_2_out,
r_in)
                                    self.dict[lst_links[1]].set_q(link_2_out + y_2_out +
r_in)
                                lst_results[cell] = self.dict[cell].n
                                lst_results_v[cell] = self.dict[cell].get_v()
                                if cell_2 != 0:
                                    lst_results[cell_2] = self.dict[cell_2].n
                                    lst_results_v[cell_2] = self.dict[cell_2].get_v()
                    results[t] = lst_results
                    results_v[t] = lst_results_v
        return TTS * self.delta_t, results, n_tot, results_v
```

```
if self.dict[cell].get_s() == True:
    if y_out == n_out:
        s = y_out * self.dict[cell].get_beta(t)
        y_out *= (1 - self.dict[cell].get_beta(t))
    else:
        s = y_out * self.dict[cell].get_beta(t)
        if s + y_out > n_out:
            ratio = n_out / (s + y_out)
            s *= ratio
            y_out *= ratio
else:
    s = 0
```

Inleiding

Het model kent drie ingebouwde intelligente verkeerssystemen. Zij proberen bij activatie filedrukte te verminderen. Alle worden besproken waarbij de programmeercode aan de rechterkant terug te zien is.

Ramp metering

RM, zoals besproken in de *Hoofdstuk 1; Bron 3*, probeert de T_s te verminderen door de voertuigen op opritten tegen te houden. Zo behoudt men de maximale intensiteit. In dit model kan RM in twee situaties worden toegepast: wanneer er een reguliere oprit is of wanneer er twee snelwegen bij elkaar invoegen. Beide maken gebruik van hetzelfde principe. Daarom is alleen het invoegen van snelwegen besproken. RM wordt bij het definiëren van de cellen op *True* of *False* gezet. Volgens wordt in de *run_ctm* voor elke tijdstap en cel gecontroleerd of RM op *True* is gezet. Tevens wordt bij het definiëren van de cellen de variabele K_r meegegeven zodat RM met een bepaalde kracht kan worden toegepast. De hoogte van K_r heeft echter weinig invloed op de resultaten van RM (zie *Hoofdstuk 1; Bron 3*).

Wanneer in een cel RM actief is, wordt voor de andere cel die verbonden is aan de ontvangende cel eerst de reguliere output berekend. Vervolgens gebruikt men ALINEA om de output vanuit de eigenlijke cel te bepalen door middel van Figuur 15 (zie *Hoofdstuk 1; Bron 3*).

$$r(k) = r(k-1) + K_r[\hat{o} - o_{out}(k)] \qquad \text{(Figuur 15)}$$

Enkel worden variabelen als o (bezetting) niet in het model gebruikt. Hierdoor is een vergelijkbare conditie geformuleerd. Hierbij wordt voor ô o_{cr} gebruikt.

$$r(t) = max\big(0, r_i(t-1) + K_r\big[K_{cr,i} * \Delta L * \mu_i - n_i\big]\big) \quad (58)$$

Daarnaast is het belangrijk om te controleren of de door de ALINEA bepaalde output niet groter is dan het aantal voertuigen in de cel. Indien dit het geval is wordt de output terugzet naar het aantal voertuigen in de cel.

Variabele snelheidslimieten (VSL)

VSL, zoals besproken is in *Hoofdstuk 1; Bron 4*, probeert de T_s te verminderen door voertuigen een maximale snelheid op te leggen. Het model is in staat een VSL te implementeren door middel van een functie genaamd VSL waaraan men een lijst met cellen meegeeft. Aan alle cellen in deze lijst wordt de meegegeven snelheidslimiet opgelegd. Vervolgens wordt er gekeken wat het effect hiervan is op de T_s.

Oorspronkelijk was deze functie een algoritme die het model honderden keren uitvoert om alle waarde van de T_s te vergelijken en hier degene met de laagste T_s te vinden. Echter, de complexiteit van het model is te hoog. Hierdoor is ervoor het runnen van veel simulaties betere hardware nodig dan de gebruikten. De functie is daarom enkel in staat om het model één keer te runnen.

Vergelijking 59:

Pythoncode voor RM

```
if cell_2 != 0:
    if self.dict[cell].RM == True and t >
self.dict[cell].RM_start / self.delta_t and t <
self.dict[cell].RM_end / self.delta_t:
        y_2_out = min(n_2_out, sum(Q_N_tot))
        y_out =  max(0, self.dict[lst_links[0]].r +
self.dict[cell].K_r*(self.dict[lst_links[0]].K_cr *
self.dict[lst_links[0]].delta_l * self.dict[lst_links[0]].lanes
- self.dict[lst_links[0]].n))
        self.dict[lst_links[0]].set_r(y_out)
        y_out = min(self.dict[cell].n, y_out)
    elif self.dict[cell_2].RM == True and t >
self.dict[cell_2].RM_start / self.delta_t and t <
self.dict[cell_2].RM_end / self.delta_t:
        y_out = min(n_out, sum(Q_N_tot))
        y_2_out =  max(0, self.dict[lst_links[0]].r +
self.dict[cell_2].K_r*(self.dict[lst_links[0]].K_cr *
self.dict[lst_links[0]].delta_l * self.dict[lst_links[0]].lanes
- self.dict[lst_links[0]].n))
        self.dict[lst_links[0]].set_r(y_2_out)
        y_2_out = min(self.dict[cell_2].n, y_2_out)
```

Vergelijking 60:

Pythoncode voor VSL

```
def VSL(self, list_cells, v_c, v_c_max, alpha, A, E):
    for cell in range(len(list_cells)):
        b = min(v_c[cell] / v_c_max[cell] * (1 +
alpha[cell]), 1)
        self.dict[list_cells[cell]].set_v_f(min(v_c_max[ce
ll] * b, self.dict[list_cells[cell]].v_f))
        self.dict[list_cells[cell]].set_K_cr(self.dict[lis
t_cells[cell]].K_cr * (1 + A[cell]*(1 - b)))
        self.dict[list_cells[cell]].set_a(self.dict[list_c
ells[cell]].a*(E[cell] - (E[cell] - 1)* b))
```

De wiskunde vergelijking die gebruikt zijn voor VSL, is identiek aan degene die is voorgesteld in het theoretisch kader (zie vergelijking 23). Dit is het meeste accurate model dat beschikbaar is en kent tevens variabelen die overeenkomen met dit model. Er wordt van lijsten gebruik gemaakt om de variabelen als de V, V_{max}, α, A en E voor elke cel individueel te definiëren.

Verhogen aantal rijbanen

Ten slotte is er een systeem geïmplementeerd die in staat in om het aantal rijbanen voor een groep cellen te verhogen. Dit is niet direct toepasbaar in een verkeerssysteem. Het geeft echter wel een idee van de mogelijke verbeteringen die gepaard gaan met het vergroten van het wegennet. In het computerprogramma vraagt men de functie *increase_lanes* op, waaraan men een lijst van cellen en de gewenste verhoging meegeeft. Vervolgens wordt in de functie het aantal rijbanen verhoogd met de gegeven verhoging. Direct daarna wordt het verkeerssysteem gesimuleerd om de gevolgen te zien.

Vergelijking 61:

Pythoncode voor Verhogen aantal Rijbanen

```
def increase_lanes(self, list_cells, increasement = 1,
start=0, end=86400):
    for cell in list_cells:
        self.dict[cell].set_lanes(self.dict[cell].lanes +
increasement)
    TTS, results, n_tot, results_v = self.run_ctm(start,
end)
    return TTS, results, n_tot, results_v
```

Voorbereiden en schatten van data

Inleiding

De data uit de open datasource van het NDW worden voorbereid voor gebruik in het model. Het oorspronkelijke format is immers niet geschikt (zie *Hoofdstuk* 2). Daarnaast moeten een aantal waarden en verhoudingen geschat worden. Dit moet nauwkeurig en op basis van ander onderzoek gebeuren. Het voorbereiden van de data is door de gevoeligheid van het model van levensbelang. Deze gevoeligheid kan eenvoudig worden geïllustreerd in een systeem van 10 cellen. Er is voor elke cel gekozen voor $v_f = 100\ (km/h)$, $Q = 1875\ (v/h)$, $K_{cr} = 47\ (v/km)$ en $K_j = 211\ (v/km)$. Ook zijn er om tijdstip nul 65 voertuigen in cel 8 geplaatst. In de werkelijkheid kan dit door bijvoorbeeld eerdere congestie zijn opgebouwd.

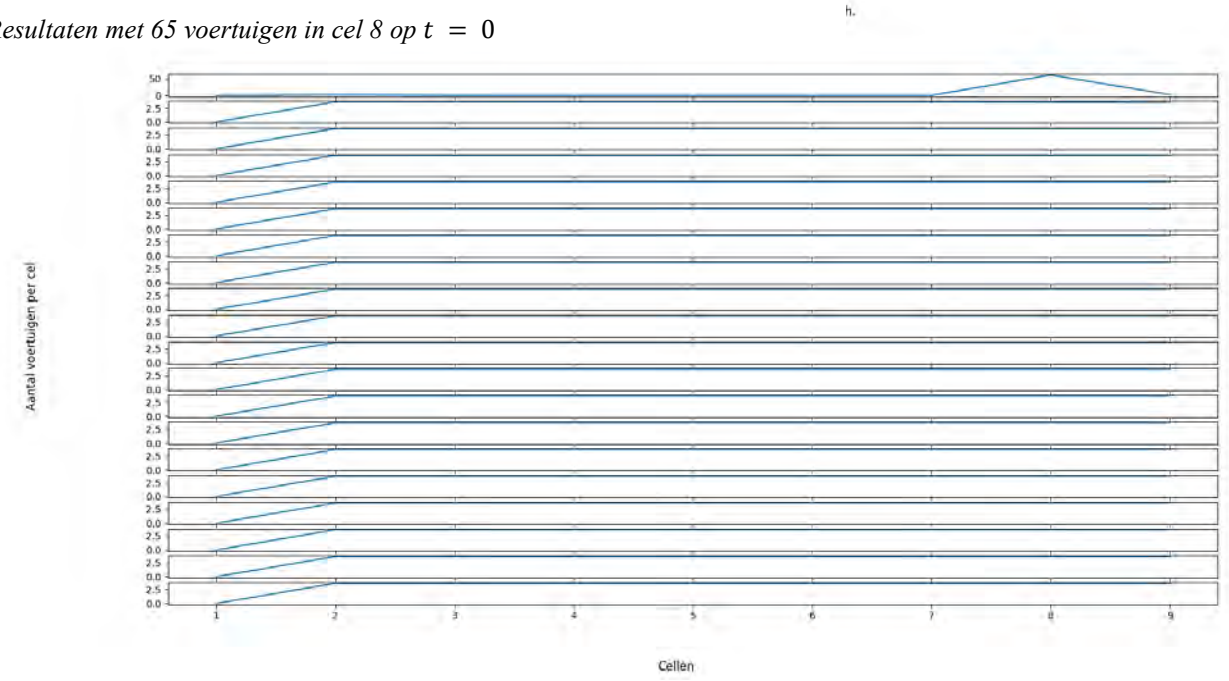
Wanneer q, vanuit de startcel, 23 uur (van de 24) op 1000 (v/h) wordt vastgesteld en 1 uur op 2500 (v/h) levert dit het volgende resultaat op:

$$TTS: \quad 3214036$$
$$N\_tot: \quad 25500$$
$$TTS/\mathrm{N\_tot}: 126$$

Hierin in *TTS* de totale bestede tijd (T_s) en *N_tot* de totale hoeveelheid voertuigen (N_{tot}) die gedurende 24 uur door het systeem gaan. Dit maakt de laatste waarde (TTS/N_tot) de gemiddelde tijd dat een voertuig zich in het systeem bevindt. Alle gegeven zijn in seconden. In figuur 26, die de hoeveelheid voertuigen in een cel op een bepaald tijdstip laat zien, is duidelijk te zien dat er spraken is van condities met een vrijestroomsnelheid door het gehele systeem. De congestie op tijdstip 0 ruimt zich snel op.

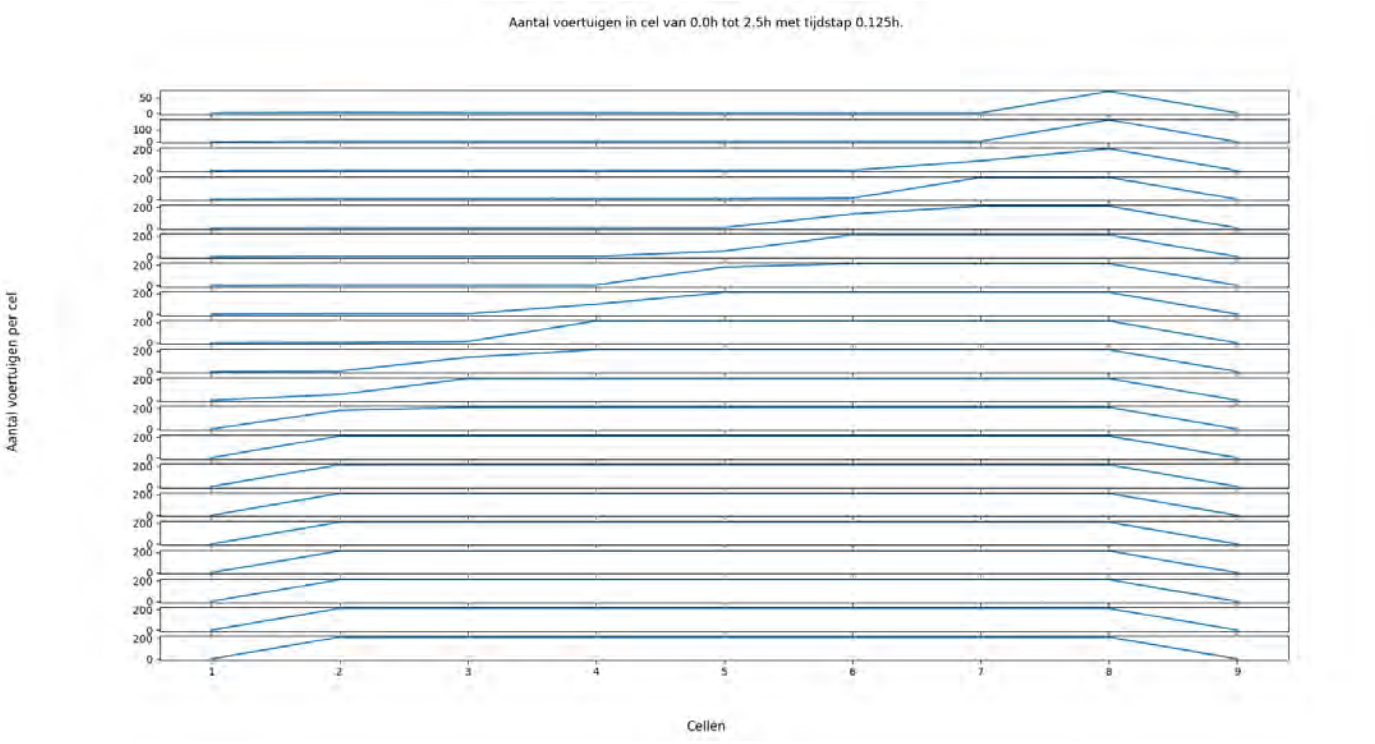
Figuur 26:

Resultaten met 65 voertuigen in cel 8 op $t = 0$



Figuur 27:

Resultaten met 70 voertuigen in cel 8 op $t = 0$



Wanneer er om tijdstip nul 70 voertuigen in cel 8 worden geplaatst levert dit echter het volgende resultaat op:

$$TTS: \quad 124032363$$
$$N\_tot: \quad 1408$$
$$TTS/N\_tot: \ 88118$$

De tijd per reiziger is verhoogd met 69832%. Er bouwt immers congestie op en snelheid verminderd naar nul. Er ontstaat een opstopping. Aangezien het model niks meegeeft, blijft de snelheid nul en zal de opstopping blijven voortduren. Dit toont de gevoeligheid van het model. Hierdoor is het van groot belang de data juist te schatten en voor te bereiden.

Het importeren en schatten van data

De data worden in een Excel bestand aangeleverd. Eerst zijn de gegevens per cel op volgende geplaatst en zijn de cellen genummerd. Wanneer het bestand wordt omgezet naar csv-bestand wordt het in het programma naar een *pandas dataframe* omgezet (een module in Python die een tweedimensionale structuur van rijen en kolommen vormt).

Volgens wordt de data ingelezen in het model. Alle gegevens worden hierbij opgeslagen in *lists* of *dictionaries* afhankelijk van de hoeveelheid en/of vorm van de data. Dit gebeurt met zogenaamde *lambda* functies. Dit proces is in vergelijking 62 te zien. N staat voor het aantal kolommen in het dataframe.

Elke cel kent een aantal belangrijke gegevens die niet in de data aanwezig zijn. Voor deze gegevens is het dus nodig om een schatting te maken zodat het model zo veel mogelijk overeenkomt met de werkelijkheid. Normaalgesproken is het mogelijk om algoritmen te runnen die kijken wat de optimale waarden zijn. Dit gebeurt door het model op veel verschillende waarden te testen. Er is echter al vastgesteld dat de gebruikte hardware dit niet aankan (zie *Hoofdstuk 4;*

Vergelijking 62:

Pythoncode van Importeren en Schatten

```
df = pd.read_csv('file')
q_data = {}
lst_Q = []
for i in range(1, N):
    q_data[i] = [float(df.iloc[j][i]) for j in range(0,24)]
lst_lambda = [df.iloc[27][i] for i in range(1,N)]
lst_delta_l = [int(df.iloc[55][i])/1000 for i in range(1,N)]
for i in range(1,N):
    q_data[i] = [float(q_data[i][j]) / int(lst_lambda[i - 1]) for j in range(0,24)]
    lst_Q.append(max(q_data[i]) * ratio)

v_data = {}
lst_v_f = []
for i in range(1, N):
    v_data[i] = [int(df.iloc[j][i]) for j in range(30,54)]
    lst_v_f.append(max(v_data[i]) * ratio)

K_data = {}
lst_K_cr = []
lst_K_j = []
for i in range(len(q_data)):
    i += 1
    K_data[i] = [q_data[i][j] / v_data[i][j] for j in range(0,24)]
for i in range(  len(lst_Q)):
    lst_K_cr.append(lst_Q[i] / lst_v_f[i] * ratio)

for i in range(len(lst_K_cr)):
    lst_K_j.append(lst_K_cr[i] * ratio)

print('v_f:  ',lst_v_f)
print('Q:    ',lst_Q)
print('K_cr: ',lst_K_cr)
print('K_j:  ',lst_K_j)
```

Werking van de optimaliseringsalgoritme). Hierdoor is men genoodzaakt deze handmatig te schatten. Het model is dus verschillende keren gesimuleerd en hierbij is de output met de output uit de eigenlijke data vergeleken en hierop aangepast. Zo zijn de waarde zo veel mogelijk in overeenkomst met de werkelijkheid. Dit geldt voor Q, v_f, K_{cr} en K_j. Voor de maximale intensiteit Q kiest men bijvoorbeeld de maximale intensiteit (q) uit de data. Deze is gegarandeerd lager dan het theoretische maximum en wordt daardoor met een bepaalde factor vermenigvuldigd. Ditzelfde is van toepassing op v_f en K_{cr}. De maximale dichtheid (K_j) is een waarde dit in verhouding staat tot (K_{cr}). Uit veel empirisch onderzoek blijkt dat vergelijking 63 en daarmee 64 van toepassing is (École Polytechnique Fédérale de Lausanne, 2022):

$$Q_i = V_{f,i} * \frac{K_{jam,i}}{4} \quad (63)$$

$$K_{jam,i} = 4 * \frac{Q_i}{v_{f,i}} \text{ ofwel } K_{jam,i} = 4 * K_{cr,i} \quad (64)$$

Daarom is deze variabele in dit onderzoek ook, in de meeste gevallen, op vier gesteld. Deze waarde worden zo gestemd dat ze overeenkomen met de gemiddelde op snelwegen. Voor Q, v_f, K_{cr} en K_j gelden respectievelijk de volgende domeinen 1800-2400 (*v/h*), 90-120 (*km/h*), 40-60 (*v/km*) en 160-240 (*v/km*) zijn.

Het visualiseren van data

Ter volledigheid heeft het model nog één functie genaamd *visualiser*. Deze zet resultaten uit het model om in een grafiek zoals deze in figuur 26 en 27 te zien is. Hierbij kan de start- en eindtijd en de tijdstap naar eigen wil worden ingesteld. De data worden omgevormd voordat het gevisualiseerd wordt. Voor de visualisatie maakt men gebruik van een ingebouwde module in Python genaamd *matplotlib*.

Complexiteit van het model

Ter analyse van het model is het van belang de complexiteit te bespreken. Zoals besproken is in *Hoofdstuk 1; Bron 1*, maakt men voor de complexiteit gebruik van de big-O-notatie. Het model kent een aantal belangrijke stappen die de big-O opmaken. Men rekent in het model voor elke tijdstap (t), voor elke cel (C) al zijn aangrenzende cellen door. Hypothetisch zijn dit er $|C| - 1$. Het aantal berekeningen dat voor één tijdstap wordt uitgevoerd is dus gelijk aan vergelijking 65. In de big-O-notatie vervallen echter hogere orde termen. Zo vormt de uiteindelijke complexiteit per tijdstap in vergelijking 66.

$$O(|C|) * [O(|C|) * O(1)] \quad (65)$$

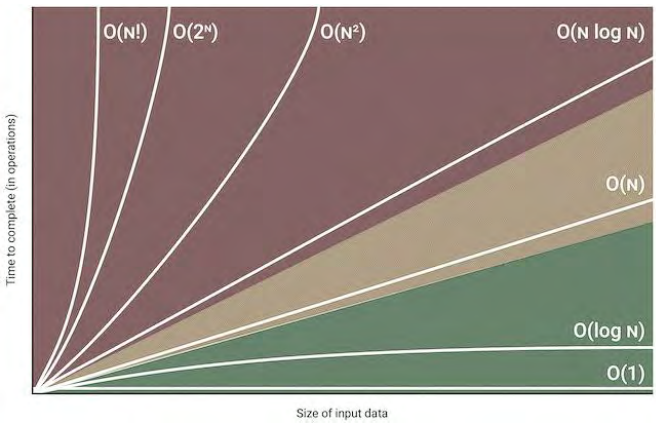
$$O(|C|) * [O(|C|) * O(1)] = O(|C|) * O(|C|)$$
$$= O(|C|^2) \quad (66)$$

Dit wordt voor elke tijdstap (ΔT) berekend. Voor de gehele *run_ctm* functie is dus vergelijking 67 van kracht.

$$O(|\Delta T|) * O(|C|^2) = O(|\Delta T| * |C|^2) \quad (67)$$

Figuur 28:

Vergelijking Complexiteiten



Noot. Miessler, D, 2019b

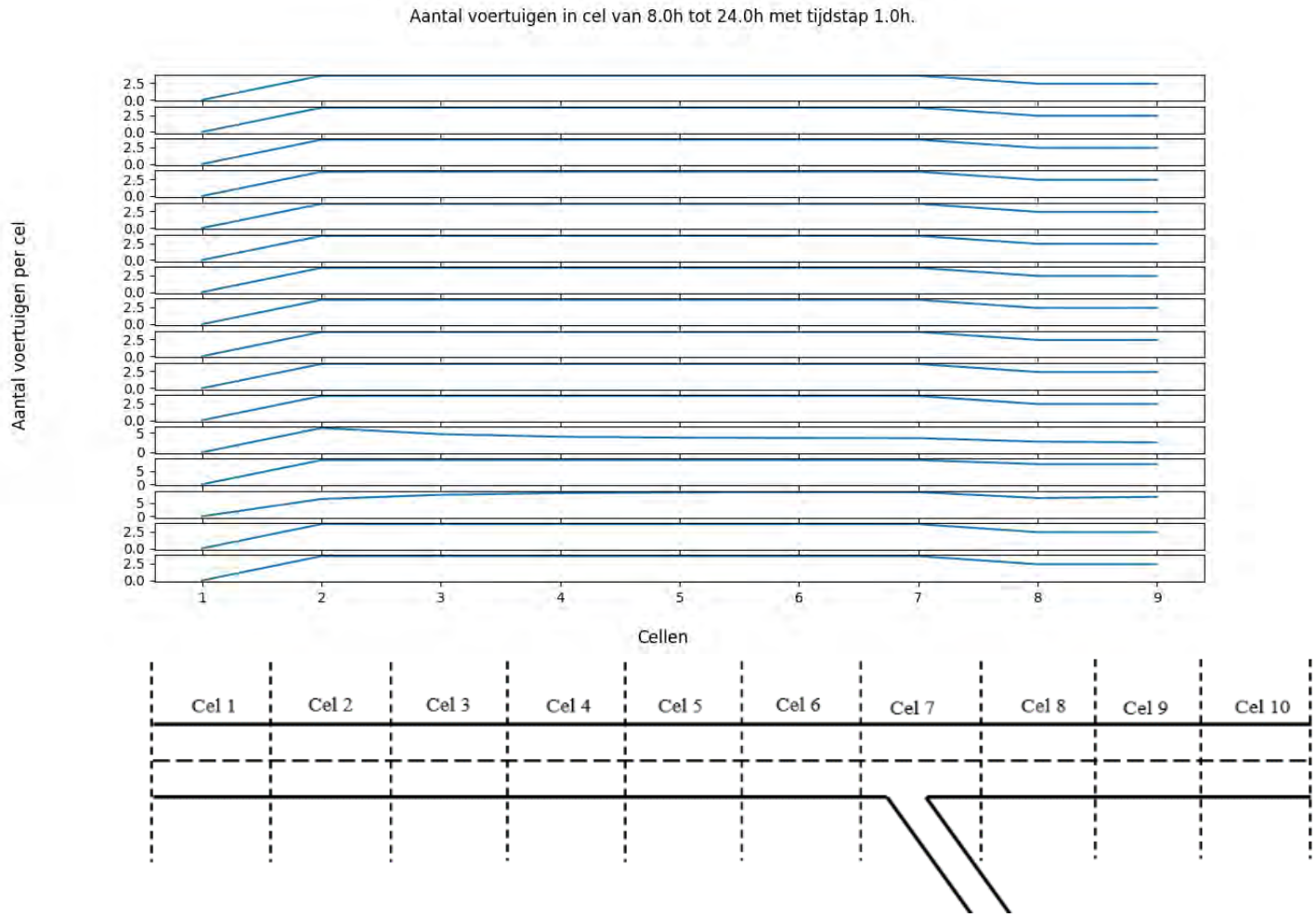
In figuur 29 zijn de soorten complexiteiten gevisualiseerd. Hierin is te zien dat de verkregen complexiteit, die vergelijkbaar, maar slechter, is dan $O(N^2)$, zich in het rode gebied bevindt. De complexiteit is dus hoog. Het is echter onmogelijk om het CTM op een andere manier te interpreteren en complexiteit te verminderen. Als gevolg moeten onderzochte verkeerssystemen relatief klein blijven om het aantal numerieke operaties laag te houden. In dit onderzoek worden dat ook een systeem met maximaal 61 cellen gebruikt (zie *Casestudie 4*) wat vervolgens over 24 uur met tijdstappen van 5 seconde wordt doorgerekend. Het aantal numerieke operaties is te zien in vergelijking 68.

$$O(|t| * |C|^2) = O\left(\frac{86400}{5} * 61^2\right) = O(6{,}4 * 10^7) \quad (68)$$

Voor het grootste systeem zijn er volgens de Big-O notatie 64 miljoen berekening nodig. Daarnaast kent het model per cel nog een groot aantal berekeningen. Deze hebben geen invloed op de complexiteit, maar verhogen wel het aantal berekeningen. Nog steeds kan een computer dit met gemak aan en daarom is de verkregen complexiteit voor dit model acceptabel.

Figuur 29:

testexperiment 1: Visualisatie model met Afrit bij cel 7



Hoofdstuk 5: testen

Voor het uitvoeren van de casestudies is het model uitvoerig getest door middel van vier testexperimenten. Vervolgens is het model aangepast indien men tegen fouten aanliep. Zo is een juiste werkingen gegarandeerd, wat de validiteit van dit onderzoek verhoogt.

De laatste cel van de verkeerssystemen is niet in de figuren zichtbaar aangezien het een outputcel is. In het model worden hier geen berekeningen voor uitgevoerd en zijn als gevolg ook geen resultaten opgeslagen die later gevisualiseerd worden.

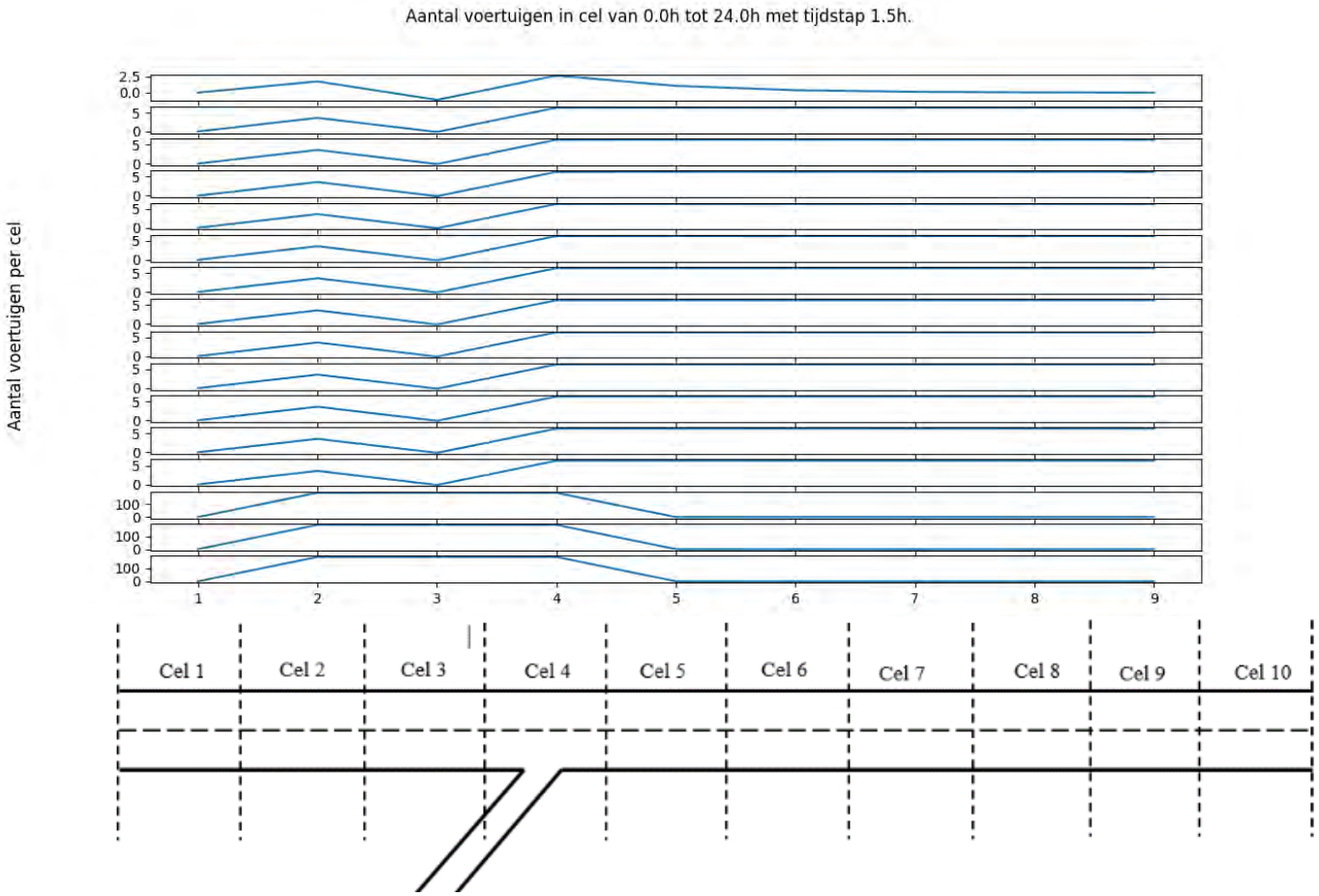
Testexperiment 1: model van 10 cellen met een afrit

Ten eerste is een testmodel gemaakt die bestaat uit 10 cellen met een afrit van $q = 500\ (v/h)$ op cel 7 (zie figuur 29). Onder omstandigheden waarbij een vrijestroomsnelheid van toepassing is, ziet men dat ervoor de afrit spraken is van eens splitsingsverhouding en dat hierdoor minder voertuigen in cel 8 en 9 zichtbaar zijn. Dit is in figuur 29 te zien. Na de eerste 12 subgrafieken is de input een aantal uur sterk verhoogd van $1000\ (v/h)$ tot $2500\ (v/h)$. Dit bevindt zich boven de maximale intensiteit. Hier maakt het model gebruik van de conditie die stelt dat de $y_{out,i}\ (t)$ gelijk mag blijven en hier een $s_i(t)$ bovenop komt (zie *Hoofdstuk 4; Werking van de simulatie*). Dit is duidelijk te zien in figuur 29. De afname van de hoeveelheid voertuigen is van cel 7 naar cel 8 minder, omdat het zich in condities met opbouwde congestie bevindt.

Dit testmodel is geslaagd. Het komt volledig overeen met de verwachte uitkomsten en hiermee met de werkelijkheid.

Figuur 30:

testexperiment 2: Visualisatie model met Oprit bij cel 4



Testexperiment 2: model van 10 celen met een oprit

Ten tweede is een testmodel gemaakt die bestaat uit 10 cellen met een oprit van $q = 1500\ (v/h)$ aan cel 4 (zie figuur 30). Dit is een onrealistisch hoge intensiteit. Echter, dit vereenvoudigt het testen van het model. Er is een constante input van 1000 (v/h) die van 19:00 tot 21:00 verhoogd is naar 2500 (v/h). Tot 19:00 kent de snelweg de vrijestroomsnelheid omdat $q_{tot} = 1000 + 1500 = 2500(v/h)$ niet de maximale intensiteit van $Q = 3000\ (v/h)$ overschrijdt (voor Q geldt in dit testexperiment $Q = q_{max} * 1{,}2 = 2500 * 1{,}2 = 3000$). Wanneer de instroom vanaf de snelweg toeneemt, in de laatste drie grafieken van figuur 31, bereikt men een q_{tot} van 4000. Dit is hoger dan de maximale intensiteit; langzaam bouwt er congestie op. Dit leidt uiteindelijke tot een opstopping omdat de snelheid afneemt tot 0 (*km/h*). Dit heeft ernstige gevolgen en het model herstelt hier niet meer van. Hier gaat het volgende resultaat mee gepaard:

$$TTS: \quad 13126866$$
$$N\_tot: \quad 19865$$
$$TTS/N\_tot: \ 660{,}8$$

Het in dit onderzoek voorgestelde model kent echter een algoritmen die in staat is verkeer te optimaliseren. Implementatie van RM houdt voertuigen op de oprit van cel 4 tegen zodat de maximale stroom wordt behouden wordt. Dit is te zien in figuur 32. Zo wordt een opstopping voorkomen. Dit leidt tot het volgende resultaat:

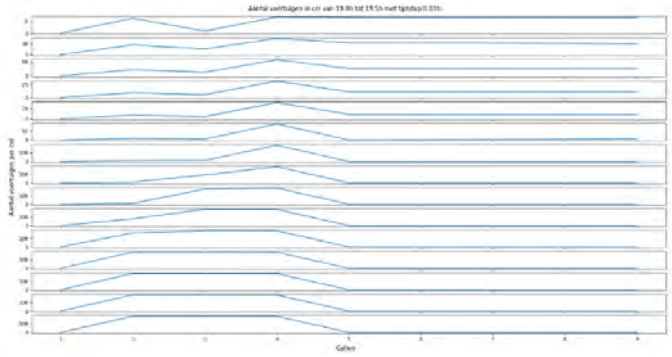
$$TTS: \quad 4592887$$
$$N\_tot: \quad 27000$$
$$TTS/N\_tot: \ 170{,}0$$

De T_s is met 186% verbeterd. De tijd per reiziger met 288%.

Dit zijn positieve resultaten die de werkingen van het model bevestigen. Enkel, het is belangrijk te beseffen dat deze waarde niet realistisch zijn. Ten eerste zijn er met overdreven waarde gewerkt. Daarnaast treedt in de praktijk een opstopping niet op. De snelheid neemt af, maar bereikt zelden een complete opstopping. Indien dit wel het geval is, is het in staat om terug te keren naar zijn oude condities als de intensiteit weer afneemt.

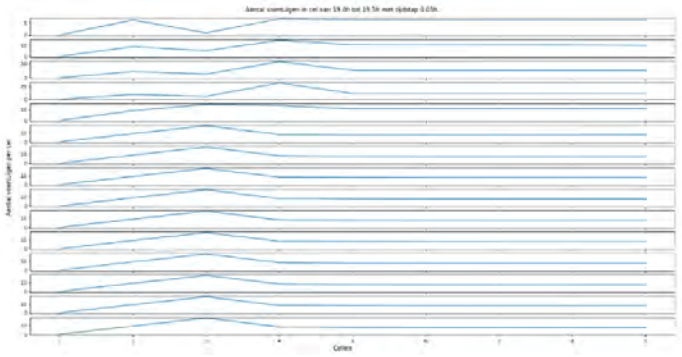
Figuur 32:

testexperiment 2: Visualisatie van opbouw Congestie na 19:00 Zonder RM



Figuur 31:

testexperiment 2: Visualisatie van opbouw Congestie na 19:00 Met RM

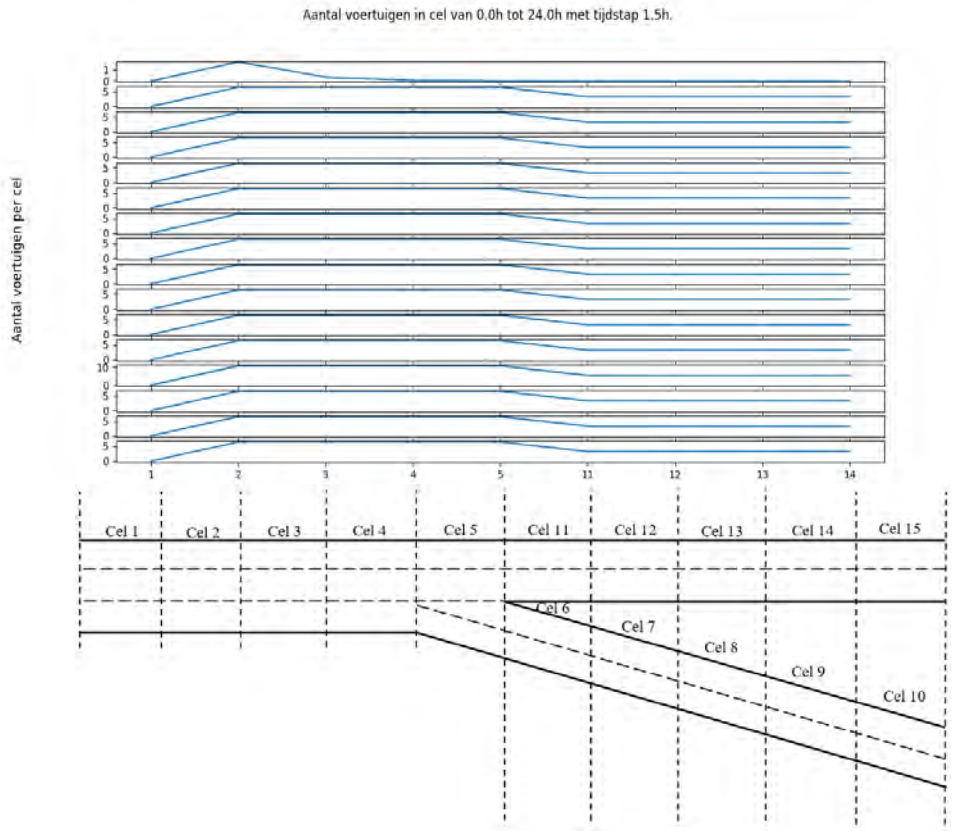


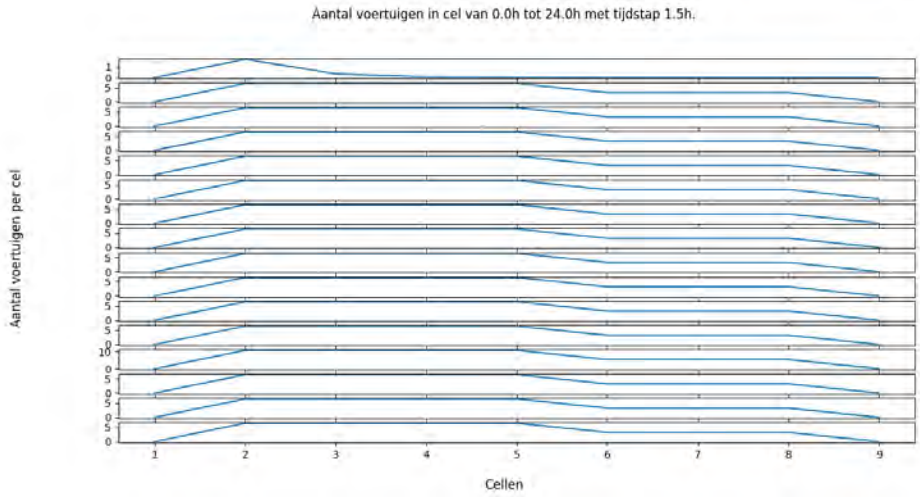
Testexperiment 3: model van 15 cellen met uitvoegende snelwegen

Het model van 10 cellen met een afrit wordt gecompliceerd door de afrit te vervangen door een uitvoegende snelwegen. 5 cellen van 3 rijbanen worden opgevolgd door 2 verschillende snelwegen van beide 5 cellen en 2 banen (zie figuur 33). De totale input is op $q = 3000\ (v/h)$ gesteld. Daarnaast is de input 2 uur lang verhoogd naar $q = 5000\ (v/h)$. De resultaten zijn in overeenstemming met situatie met een afrit. Dit laat zien dat ook dit verkeerssysteem succesvol is geïmplementeerd in.

Figuur 33:

test experiment 3: Visualisatie model met Uitvoegende Snelwegen in cel 5





Testexperiment 4: model van 15 cellen met invoegende snelwegen

Na een model van 10 cellen met een oprit wordt er gekeken naar een model waarbij twee snelwegen bij elkaar invoegen. De opbouw van het systeem is het spiegelbeeld van testexperiment 3. De gekozen waarde van het model zijn vergelijkbaar aan een testexperiment 2. De andere snelweg wordt nu alleen ook beïnvloed door de simulatie. Dit maakt het ingewikkelder. De resultaten laten zien dat de hoeveelheid voertuigen in de cel verdubbeld wanneer de snelwegen bij elkaar invoegen. Dit levert het volgende resultaat op:

$$TTS: \quad 11224795$$
$$N\_tot: \quad 78000$$
$$TTS/N\_tot: \ 143{,}9$$

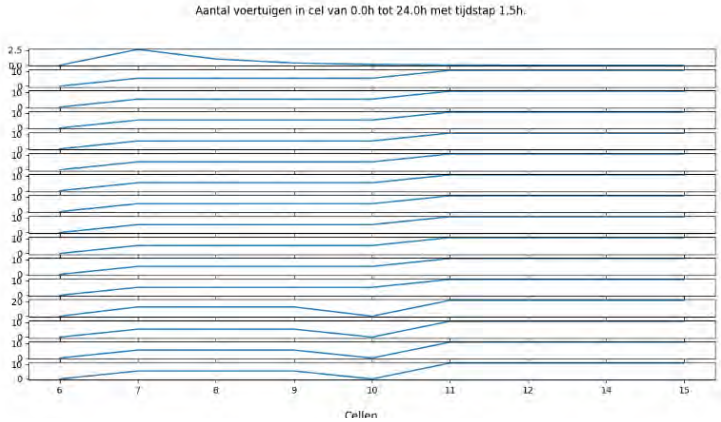
Tevens is RM op dit testscenario toegepast. Dit is hier gedaan voor cel 5 (deze heeft een lagere input dan de reeks met cel 10 en is daarom het meest geschikt voor RM). Dit is te zien in figuur 34 en 35. Het leidt tot het volgende resultaat:

$$TTS: \quad 11000410$$
$$N\_tot: \quad 78000$$
$$TTS/N\_tot: 141{,}0$$

Dit T_s verbeterd hier met 2,00%. Dit is een kleine verbeteringen ten opzichte van de eerdere test. Doordat er met ruime schattingen gewerkt is, is het testexperiment 3 naar boven uitgevallen terwijl testexperiment 4 naar beneden uitvalt. Toch tonen de resultaten dat model juist functioneert.

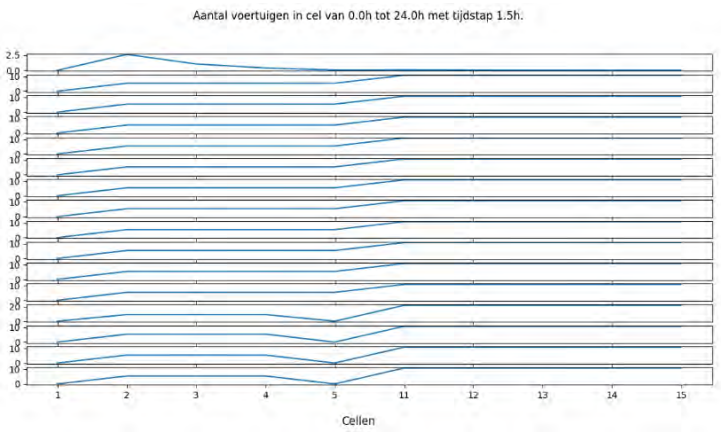
Figuur 34:

testexperiment 4: Effecten van RM op cellen 1 t/m 5



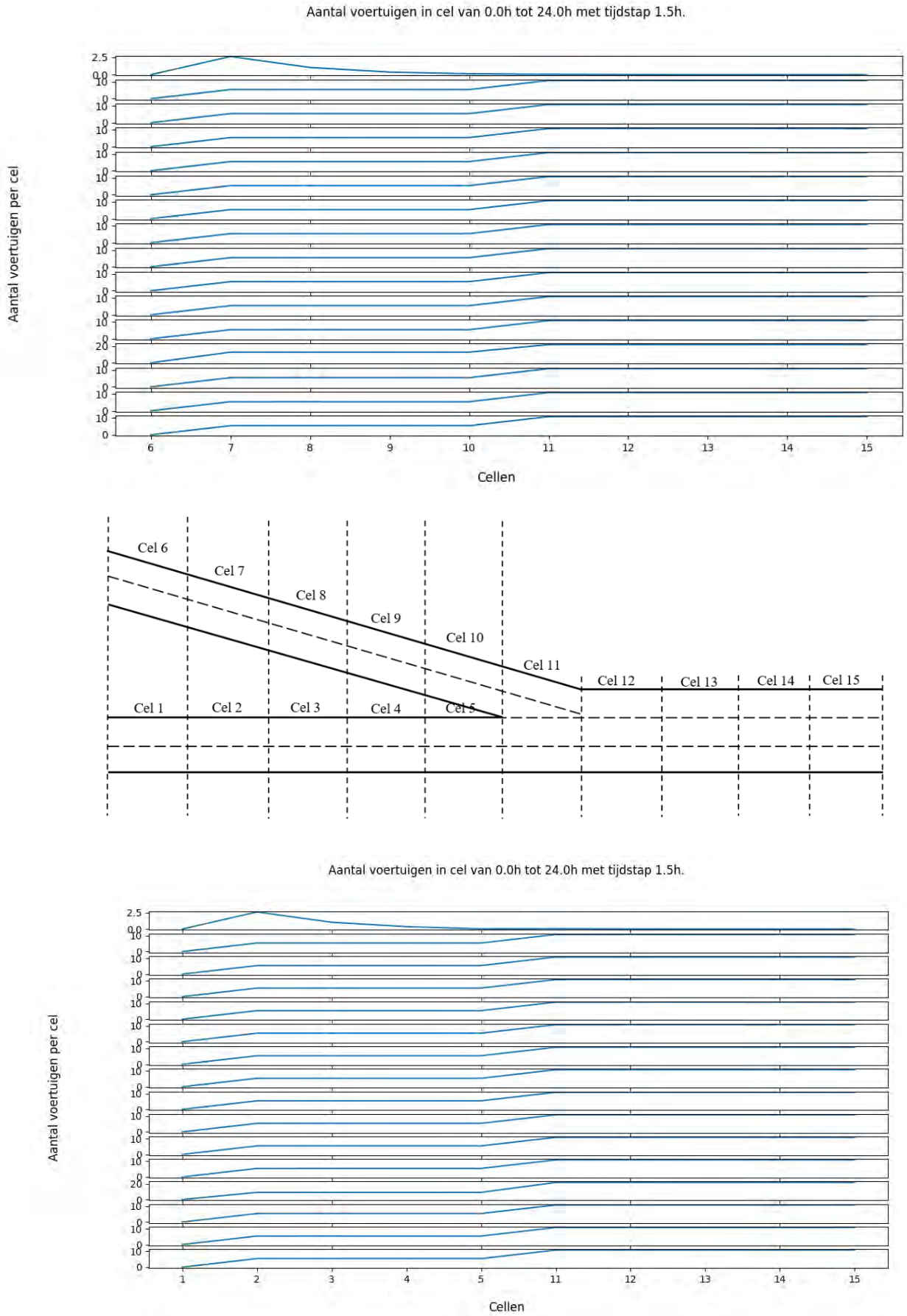
Figuur 35:

testexperiment 4: Effecten van RM op cellen 6 t/m 10



Figuur 36:

testexperiment 4: Visualisatie model met Uitvoegende Snelwegen in cel 11



Hoofdstuk 6: onderzoek

In het onderzoek zijn vier casestudies uitgevoerd. Telkens is eerst de keuze voor het gebied uitgelegd. Daarna is de huidige situatie geanalyseerd. Vervolgens is de implementatie in het model laten zien en ten slotte zijn er verbeteringen aangeschreven indien deze mogelijk zijn.

De gekozen periode is van 07/06/2021 tot 11/06/2021. Dit zijn vijf werkdagen buiten de schoolvakanties en voldoet hiermee aan de eis uit in *Hoofdstuk 2*. Naast dat dit een gemiddelde periode, is die een beeld geeft over situatie het gehele jaar, is het een periode waarin van alle detectielussen de data beschikbaar is. Hierdoor is ervoor 2021 gekozen in plaats van voor 2022. Dit terwijl 2022 op papier een betere keuze is. 40 jaar moest men rekening houden met autoloze zondagen, vandaag kent men andere problemen: corona. Lockdowns en avondklokken zorgde voor lege snelwegen en de niet alleen de filedrukte, maar ook de luchtvervuiling nam af (Ministerie van Infrastructuur en Waterstaat, 2021). Enkel levert dit geen bruikbare gegevens voor dit onderzoek en worden de desbetreffende data niet gekozen. Gelukkig ging Nederland weer ‘open’ op 05/06/2021 en is ervan een door corona beïnvloeden situatie geen spraken (RIVM, 2022). Voor de programmeercode en gebruikte data moet men *Bijlage 4* raadplegen.

Na elke casestudie volgt, onder de subtitel *conclusie*, een conclusie over de gehele casestudie, dit om onnodige complexiteit in de uiteindelijke conclusie, veroorzaakt door verwijzingen, te voorkomen. Ook worden de resultaten vergeleken met de resultaten van de empirische data onder de subtitel *vergelijking van resultaten*.

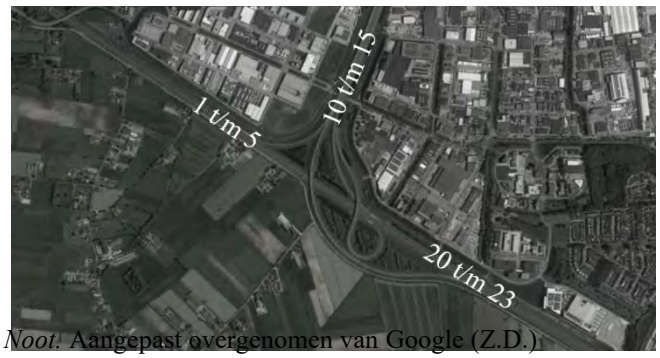
Figuur 38:

Detectielussen Knooppunt Maanderbroek

Noot. NDW, 2022

Figuur 39:

Satellietfoto Knooppunt Maanderbroek



Noot. Aangepast overgenomen van Google (Z.D.)

Figuur 37:

Lege Snelwegen door Corona

Noot. Veilig Verkeer Nederland (2020)

Casestudie 1: knooppunt Maanderbroek (A12/A30)

Inleiding

Knooppunt Maanderbroek is een minder bekend knooppunt omdat het geen knelpunt in ons wegennet is (enkel bij incidenten) en het dus weinig op bijvoorbeeld de radio te horen is. Toch is het voor een eerste casestudie geschikt. Het komt grotendeels overeen met de eisen gesteld aan een casestudie in *Hoofdstuk 2; Het onderzoeksgebied*. Enkel, het voldoet niet aan de eis ‘congestie’. Dit is bewust gedaan. Voor de eerste casestudie is het belangrijk om met een relatief eenvoudig knooppunt te werken. Het invoeren van een knooppunt en het voorbereiden van de data is niet eenvoudig. Bij systeem van dit kaliber zijn fouten nog eenvoudig herkenbaar en te verhelpen. Bij ingewikkelde modellen met veel congestie is dit niet meer mogelijk. Daarom kennen de casestudies een opbouwende graad aan complexiteit.

Beschrijving

Knooppunt Maanderbroek is een aangepast trompetknooppunt doordat de aansluiting van de A30 op de zuidoostelijke A12 de prioriteit krijgt (zie figuur 38 en 39). Het kent daarom een fly-over in plaats van een klaverblad. De A30 is een tweebaansweg; de noordwestelijke A12 is een driebaansweg. Deze twee voegen zich samen tot de vierbaans zuidwestelijke A12. Door de prioriteit van de zuidoostelijke richting is in deze casestudie alleen deze verkeersrichting geanalyseerd. Dit betekent dat het vanuit deze richting een systeem is van twee invoegen snelwegen (met verschillende op- en afritten). De gebruikte detectielussen kenden geen storingen gedurende de opgevraagde periode (NDW, 2022).

Modellering

Het model maakt gebruik van de data van 19 detectielussen. Die een cel, oprit of afrit representeren. Gemiddeld is een cel 500 meter lang en de minimale lengte is 350 (m). Het kent een vrijestroomsnelheid van maximaal 121 (km/h). Hierdoor voldoet het met een tijdstap van vijf 5,0 (s) aan de eerder vastgestelde CFL-conditie (zie *Hoofdstuk 1; Bron 5* en Hoofdstuk *4*).

De opbouw van het systeem is vereenvoudigd in figuur 39. Van de detectielussen zijn er 9 aanwezig op de A30 (inclusief 2 opritten en 1 afrit). Dit zijn in het model de cellen 10 t/m 15. Daarnaast bestaat het deel van A12 in noordwestelijke richting uit 6 detectielussen (inclusief 1 oprit). Dit zijn cellen 1 t/m 5. Ten slotte bestaat het deel op de A12 in zuidoostelijke richting uit 4 detectielussen. Dit zijn cellen 20 t/m 23. Dit zijn relatief weinig cellen. Een mogelijke cel 24 kent echter een knooppunt met de N781. Dit is niet relevant tot dit onderzoek en daardoor is ervoor maar 4 cellen gekozen.

Voor de gebruikte verhoudingen van Q, v_f, K_{cr} en K_j ten opzichte van de empirische data geldt respectievelijk 1,3; 1,0; 3,5 en 4,0. Hierdoor vallen alle gegevens binnen de gestelde limieten (zie *Hoofdstuk 4*).

Resultaten

De data worden gekenmerkt door een verkeersverloop die een naast een ochtend- en avondspits tussendoor ook een redelijk hoge intensiteit kent. Dit is duidelijk in het de resultaten te zien (zie figuur 40 en 41). 's Ochtends is er immers een kleine piek te zien. 's Avonds is er een avondspits zichtbaar. De intensiteit veranderd hier op de A12 (uit noordoostelijke richting) van $q \approx 2500$ (v/h) om 14:00 naar van $q \approx 4000$ (v/h) om 17:00 uur. Deze piek is zowel figuur 40 als 41 te zien.

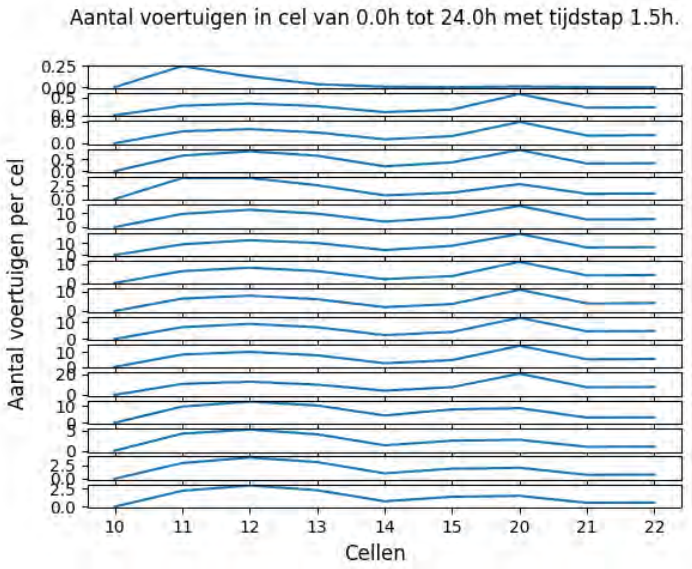
Enkel kan de A12 dit in deze simulatie niet aan. Er ontstaat immers een opstopping (die de hele simulatie voortduurt). Dit is te verklaren doordat de startcel in (te) hoog tempo voertuigen overplaatst naar cel 2. Cel 2 verplaatst deze voertuigen echter niet snel genoeg verder naar cel 3. Hierdoor bouwt er congestie op. De snelheid neemt af naar 0 wat een opstopping als gevolg heeft. Waarom cel 2 niet snel genoeg voertuigen output is te verklaren door de maximale intensiteit (Q). Voor cel $1,\ 2,\ 3,\ 4, 5$ geldt respectievelijk:

$$[1873, 2108, 1306, 2065, 1552]$$

De intensiteit van 1552 (v/h) is te verklaren door de aanwezigheid van een afrit in de vorige cel, die de intensiteit opsplitst. De maximale intensiteit van cel 3 is echter alleen door een meetfout te verklaren. Wanneer men een aanpassen maakt en het stelt op het gemiddelde van cel 2 en 4 (2086,5 (v/h)) verplaatst ditzelfde probleem zich naar cel 20. De maximale intensiteit van cel 21 wijkt namelijk ook af en is daarom aangepast. Dit kan alleen verklaard worden door een afwijking van de detectielus.

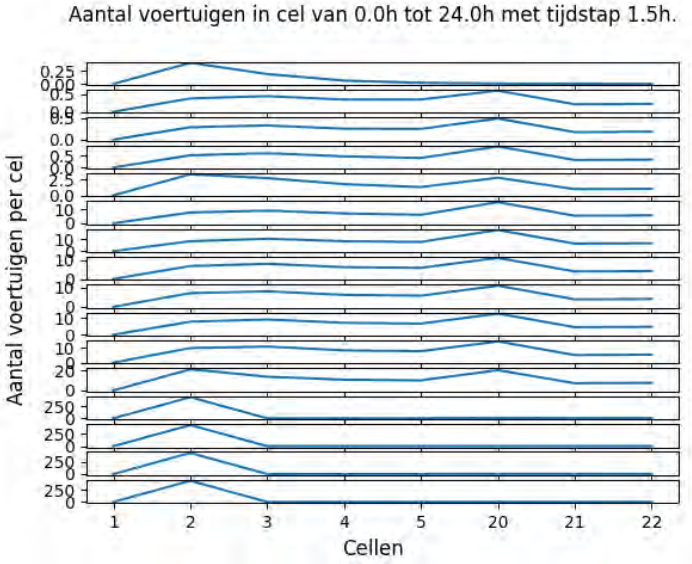
Figuur 40:

Visualisatie model A12/A30 Zonder Aanpassingen



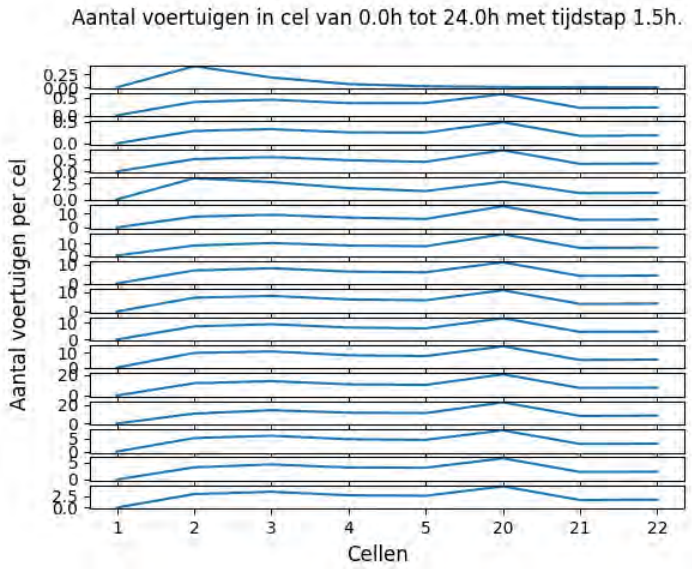
Figuur 41:

Visualisatie model A12/A30 Zonder Aanpassingen



Figuur 42:

Visualisatie model A12/A30 Na Aanpassing



Zonder deze aanpassingen is het resultaat:

$$TTS: \quad 16396804$$
$$N\_tot: \quad 64129$$
$$TTS/N\_tot: \ 255{,}6$$

Deze is verbeterd tot:

$$TTS: \quad 6772919$$
$$N\_tot: \quad 75365$$
$$TTS/N\_tot: \ 89{,}9$$

Optimalisering

Aangezien de empirische data geen congestie kent en ook de simulatie omstandigheden met een vrijestroomsnelheid kent, is er geen optimalisatie mogelijk.

Vergelijking van resultaten

De functionering van het model wordt bepaald door de verkregen snelheden te vergelijken met de oorspronkelijke snelheden. Dit wordt gevisualiseerd met nog niet eerder geïntroduceerde grafieken: degene van de snelheid. In figuur 43 en 44 ziet men dat het model sterk vereenvoudigd is ten opzichte van de werkelijkheid. Toch komt het overeen en kennen ze beide dezelfde karakteristieken. Men herkent in empirische data echter de aanwezigheid van de ochtend- en avondspits, terwijl dat deze in het de snelheden verkregen uit de simulatie niet zichtbaar zijn. Dit wordt veroorzaakt door vereenvoudigingen, die hierdoor sommige spitsen niet opmerken. Het verschil is uitgedrukt in de gemiddelde afwijking. Hiervoor wordt voor elke tijdstip en elke cel het verschil genomen tussen de snelheid van het model en van de data, volgens worden al deze waarde gesommeerd en wordt ten slotte het gemiddelde hiervan genomen. Oftewel:

$$gemiddelde\ afwijking = \frac{\sum_t \sum_i |v_{model,i}(\mathrm{t}) - v_{data,i}(\mathrm{t})|}{|\Delta T| * |i|} \quad (69)$$

Hierin is $v_i(t)$ de snelheid op een bepaald tijdstip t in een bepaalde cel i, $|\Delta T|$ het totaal aantal tijdstappen en $|i|$ het totaal aantal cellen. Ditzelfde is, geprogrammeerd in Python, in vergelijking 70 te zien. Wanneer dit wordt toegepast op casestudie 1 is de gemiddelde afwijking 10,59 (*km/h*). Dit laat zien dat het model redelijk accuraat is, maar door de vereenvoudigingen afwijkt.

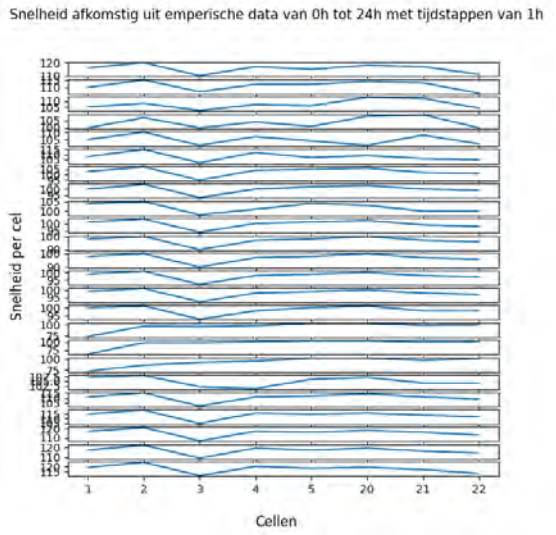
Vergelijking 70:

Pythoncode berekenen Gemiddelde Afwijking

```
def average_difference(results_1, results_2):
    print(results_1, results_2)
    z = 0
    for t in range(24):
        for i in range(len(results_1[1])):
            z += abs(results_1[t][i] - results_2[t][i])
    z /= 24 * len(results_1[1])
    print(z)
```

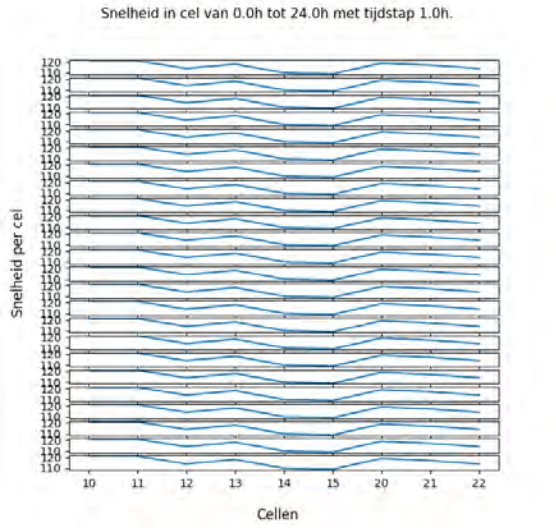
Figuur 43:

Visualisatie van Snelheid afkomstig van Data A12/A30



Figuur 44:

Visualisatie van Snelheid afkomstig van Model A12/A30



Conclusie

Het model laat realistische resultaten zien met een gemiddelde afwijking van 10,59 (*km/h*). Ter conclusie, het model brengt duidelijkheid over het gebruik van empirische data. Dit kan succesvol worden toegepast op het model, enkel moet er rekeningen gehouden worden met fouten in de meetgegevens die voor opstoppingen zorgen en kunnen de vereenvoudigingen die het model maakt voor afwijkingen zorgen.

Casestudie 2: aansluiting Barneveld (A1/A30)

Inleiding

De samenkomst van de A1 en de A30, ook wel aansluiting Barneveld, is een klassieker op het gebied van filedrukte. Er is nagenoeg dagelijks file aanwezig. Nieuwe plannen voor het maken van een fly-over, vanaf de A30 naar de A1 in westelijke richting, liggen dan ook op tafel (MIRT, Gemeente Barneveld, 2018). Echter, dat zijn langetermijnverbeteringen. Dit onderzoek kijkt vooral naar mogelijke kortetermijnverbeteringen door het implementeren van de optimaliserende algoritmen. Toch wordt er dit onderzoek ook een schatting van langetermijnoplossingen gedaan. Dit knooppunt is geschikt voor dit onderzoek. Het kent namelijk veel congestie en detectielussen. Daarnaast is dit systeem onafhankelijk. Er zijn naast het knooppunt Hoevelaken op ongeveer 10 kilometer geen andere intersecties te vinden. Daarnaast kent het een hogere complexiteit dat de vorige casestudie. Het kent immers meer en complexere op- en afritten.

Beschrijving

Het Knooppunt A1/A30 is een aangepast half klaverblad. Het kent namelijk ook input vanuit de N301. Dit resulteert over het algemeen tot een verhoging van de congestie. Er is immers een kruispunt aanwezig op een snelweg. De prioriteit ligt in westelijke richting. Dit is ook de kant waar de belangrijkste fly-over geplaatst gaat worden tafel (MIRT, Gemeente Barneveld, 2018). Daarom is dit de richting waarover de simulatie plaatsvindt. De A30 is een tweebaansweg. De oostelijke A1 is een tweebaansweg. Samen voegen ze in tot de westelijke A1 die een tweebaansweg is. Enkel gaat dit vooraf aan een lange oprit (van drie rijbanen). De data voor de gewenste periode was in zijn geheel beschikbaar. De gebruikte detectielussen kenden geen storingen in de opgevraagde periode (NDW, 2022).

Figuur 45:

Satellietfoto Knooppunt A1/A30

Noot. Aangepast overgenomen van Google (Z.D.)

Figuur 47:

Detectielussen Knooppunt A1/A30

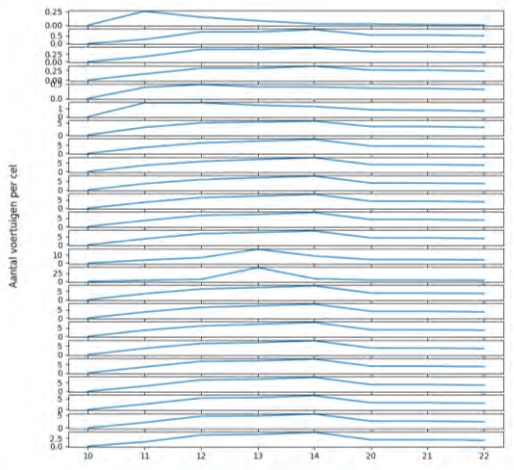
Noot. NDW (2022)

Modellering

Het model maakt gebruik van 15 detectielussen. De minimale lengte van en cel is 300 (m) met een maximale snelheid van 118 (km/h). Hierdoor voldoet de gekozen tijdsstap aan de CFL-conditie (zie *Hoofdstuk 1; Bron 5* en *Hoofdstuk 4*). Van de detectielussen zijn er 7 (inclusief 1 oprit en 1 afrit) aanwezig op de A30, deze vormen cel 10 t/m 14. 4 detectielussen vormen de oostelijke A1 (inclusief 1 afrit). Dit zijn cel 1 t/m 3. Dit is een klein aantal, anders is de af- en oprit 16 (Voorthuizen) in het model verwerkt. Iets gelijks geldt voor de westelijke A1 die bestaat uit maar 4 detectielussen. Dit zijn cel 20 t/m 23. Voor de gebruikte verhoudingen van Q, v_f, K_{cr} en K_{jam} ten opzichte van de empirische data geldt respectievelijk 1,2; 1,0; 2,3 en 4,0. Hierdoor vallen alle gegevens binnen de gestelde limieten. Het kruispunt dat dit systeem simuleert, brengt daarentegen moeilijkheden met zich mee. De heterogene condities, waarvan hierbij spraken is, gaat ver buiten het niveau van dit onderzoek. Deze zorgen in de realiteit echter voor veel ophoping in het systeem. Daarom is besloten om alle intensiteiten met een factor van 1,75 te vermenigvuldigen. Op geen andere manier zijn de condities op de snelweg te realiseren. Het kent anders constant een vrijestroomsnelheid. Dit is niet realistisch. (Deze verhoging van q is achteraf ingesteld en heeft daardoor geen invloed op Q, v_f, K_{cr} en K_{jam}.)

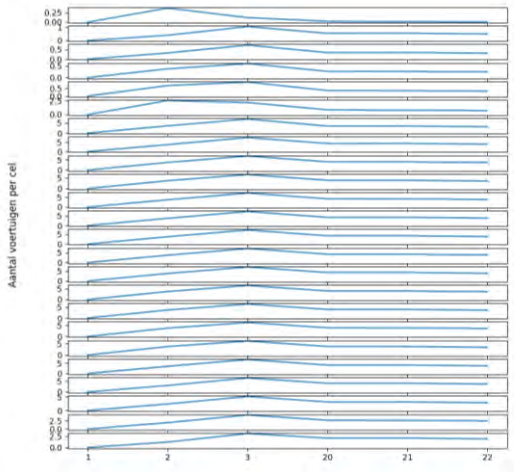
Figuur 48:

Visualisatie model A1/A30



Figuur 46:

Visualisatie model A1/A30



Ten slotte is Q van cel 11 verlaagt tot 932,9 (v/h). Men loopt hier tegen een systematische fout van het model aan. Wanneer de cel $n + 1$ na de startcel n een (buitengewoon) hoge waarde voor Q kent, output de startcel veel voertuigen naar cel $n + 1$. Indien de hierop volgende cel $n + 2$ een lagere Q kent, krijgt deze een onnatuurlijk aantal voertuigen binnen uit cel $n + 1$. Hierdoor bouwt er onjuiste congestie op. Dit leidt tot een opstopping die blijft voortduren. Het is probleem dat noch door de data, noch door theorie gevormd wordt. Het ligt enkel aan de vereenvoudigingen van het model. Het heeft echter desastreuze gevolgen en wordt opgelost door Q van cel $n + 1$ aan te passen. Er is hierbij telkens naar een waarde gestreefd die opstoppingen voorkomt en tegelijkertijd (zo veel mogelijk van de) congestie behoudt. Dit verklaart dat de waarde erg specifiek zijn. Men komt dit probleem in complexere modellen niet tegen omdat cellen, voorafgaand aan de cel met hoge waarde voor Q, een lagere waarde voor Q of betere distributie kennen waardoor de cel met een lage waarde voor Q nooit zo veel voertuigen binnenkrijgt.

Resultaten

De intensiteitsdata is vergelijkbaar met casestudie 1. De intensiteit is de hele dag relatief hoog. Wel is het opvallend dat de A1 over de hele dag homogener verdeeld is. De intensiteit op de A30 kent echter een ochtend- en avondspits. Dit is ook duidelijk in de resultaten terug te zien waar men congestievorming op de A30 (in cel 13) begint te herkennen gedurende de avondspits. Congestie bouwt zich in cel 13 op doordat tot en met hier alle cellen tweebaansweg zijn geweest en het hier voor de laatste cel wijzigt naar een eenbaansweg. (Dit is dus een verbeterpunt). Daarnaast is het opvallend om te zien dat de A30, die een lagere intensiteit heeft, op elke tijdstap meer voertuigen in de cellen heeft vergeleken met de A1. Dit is verklaren doordat de A30 een significant lagere vrijestroomsnelheid heeft. Voertuigen verblijven dus langer in een cel. De v_f zakt bijvoorbeeld in cel 13 naar wel 48 (*km/h*) (dit speelt dus mee in de congestie die zich in de cel 13 vormt). Wanneer de v_f in deze cellen verhoogd wordt, brengt dit vanzelfsprekend een verbeteringen met zich mee. Het huidige resultaat is:

$$TTS: \quad 4324196$$
$$N\_tot: \quad 72937$$
$$TTS/N\_tot: 59{,}3$$

Optimalisering

Ramp metering

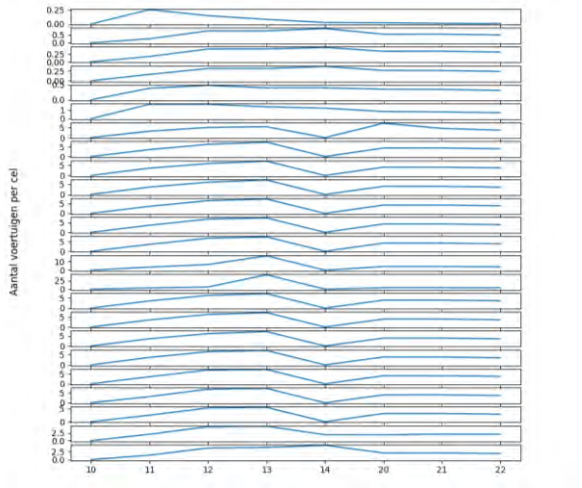
Op de korte termijn zijn er verschillende verbeteringen mogelijk. Ten eerste zijn de effecten RM onderzocht. Indien RM toegepast wordt van 06:00 tot 22:00 op cel 14 geeft dit een resultaat van:

$$TTS: \quad 3867488$$
$$N\_tot: \quad 72938$$
$$TTS/N\_tot: 53{,}0$$

Hier veranderd de T_S procentueel met $(-)10{,}6\%$. RM is in de praktijk gemakkelijk toepasbaar omdat de A30 een lange 'oprit' vormt met een hoge capaciteit waar de opbouw van congestie geen probleem is.

Figuur 49:

Visualisatie model A1/A30 Met RM



Figuur 50:

RM op A30

Noot. XON (2021)

Variabele snelheidslimieten

Naast RM beschikt het in dit onderzoek voorgestelde model over VSL. In deze casestudie is VSL toegepast op cel 10 t/m 14. Dit zijn immers de cellen waar het meeste congestie optreedt. Daarnaast heerst hier de grootste hoeveelheid aan heterogene condities. VSL staat bekend als een techniek die heterogene condities immers verminderd (zie *hoofdstuk 1; bron 4*). VSL is geïmplementeerd in de avondspits (d.w.z. 16:00-19:00). Het is hier gesteld op 40 (*km/h*) met een naleving van 60%. Dit leidt tot een v_f van 64 (*km/h*). Dit levert het volgende resultaat op:

$$TTS: \quad 4621956$$
$$N\_tot: \quad 72936$$
$$TTS/N\_tot: 63{,}4$$

Dit is een afname van de T_S van 6,91 %. Toch moet een ander belangrijk aspect niet vergeten worden. De grotere homogeniteit die gepaard gaat met een egalere snelheid is immers een voordeel (zie figuur 51). Dit heeft een positief effect op de rijstijl van bestuurders die minder hoeven te anticiperen op verkeerssituatie. VSL verlaagt dus op een andere manier mogelijk de T_S. Tevens verkleint VSL de kans op ongevallen (zie *hoofdstuk 1; bron 4*). Enkel vallen hier geen harde conclusies over te trekken omdat de homogeniteit van bestuurders en het effect hiervan op het gedrag van bestuurders niet in het model geïncorpereerd is. Dit gaat immers ver buiten de scope van dit onderzoek. Echter, in het

begin van de deze casestudie is q wel met een bepaalde factor verhoogd om op de heterogene condities na te bootsen. Door VSL verminderen worden deze condities homogeneren waardoor q mogelijk verlaagd wordt, dit een aanzienlijke verbetering van de T_s met zich meebrengen. De grootte van deze vermindering valt niet met dit model te bepalen.

Langetermijnverbeteringen

Ook zijn er mogelijke langetermijnverbeteringen waarbij men gebruik maakt van schattingen wat minder met de werkelijkheid overeenkomt.

Het voornaamste probleem van dit knooppunt is een kruispunt te midden ervan. Hierdoor wordt snelheid verlaagt en zorgt het voor de opbouw van congestie. Daarnaast gaat de snelweg kort terug naar een eenbaansweg. Wanneer dit beide wordt verholpen door de desbetreffende cellen aan te passen naar 'normale' condities van $v_f = 110$ en $\lambda = 2$, levert dit het volgende resultaat:

$$TTS: \quad 3399398$$
$$N\_tot: \quad 72937$$
$$TTS/N\_tot: 46{,}6$$

Hierbij is spraken van een verbetering van de T_s van 21,4%. Enkel zijn er in de realiteit nog grotere verbeteringen doordat de homogeniteit onder bestuurders sterk toeneemt. Dit betekent dat de verhoging van q komt te vervallen. Dit is alleen niet op een realistische manier in het model te incorporeren. Deze verbeteringen worden bewerkstelligd door bijvoorbeeld het implementeren van een fly-over.

Vergelijking van resultaten

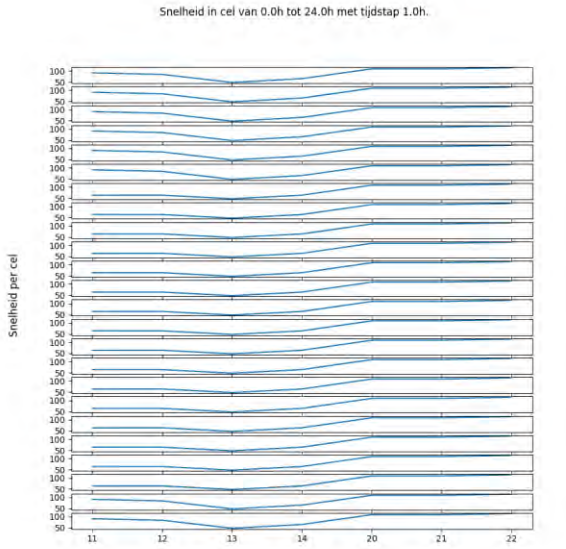
Tijdens voornamelijk spitsuren zakt de snelheid terug tot bijna een opstopping, dit is te zien in figuur 52 en 53. Het in dit onderzoek gepresenteerde model geeft wel congestie, maar in een minder mate. Mogelijk wordt dit verklaard door het feit dat de startcellen te dicht bij het knooppunt zijn gekozen. Hierdoor is in deze cellen file (en dus een lage intensiteit). Als men een desbetreffende intensiteit zonder omstandigheden met congestie in het systeem brengt, loopt het model juist vloeiend. Dit volgt uit de eerder besproken paradox (zie *Hoofdstuk 1; Bron 2*). Het verschil in snelheid bedraagt nu 23,78 (km/h).

Conclusie

Het verloop van het model is juist. Echter, zelfs met de vereiste aanpassing van de intensiteit q vertoont het model ondermaatse hoeveelheden aan congestie. De afwijking van de snelheid is daarom 23,78 (km/h). VSL verhoogt de T_s met 6,91 %. RM verlaagt de T_s met 10,6 %. Dit toont de werking van vooral RM aan. RM is immers door Rijkswaterstaat daadwerkelijk toegepast op het knooppunt (zie figuur 50). De langetermijnverbeteringen is 21,4%. Hiermee is wederom laten zien dat het voorgestelde model in resultaat overeenkomt met grote bedrijven of Rijkswaterstaat. Het toont duidelijk dat het implementeren van een fly-over in de toekomst een mogelijkheden is (MIRT, Gemeente Barneveld, 2018).

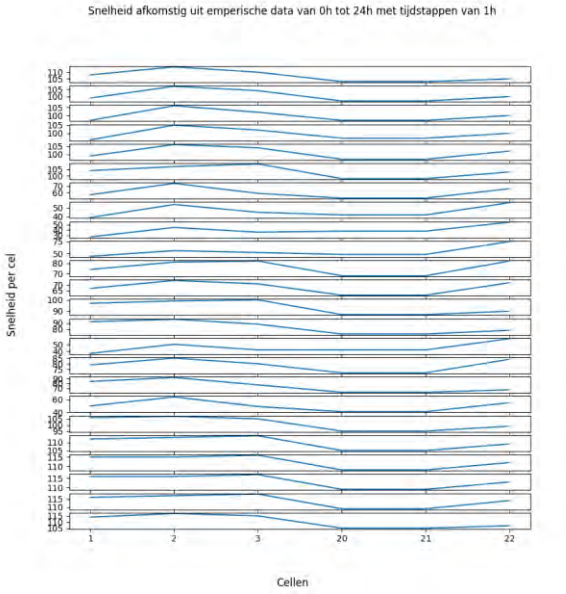
Figuur 51:

Effecten van VSL op de Homogeniteit van de Snelheid



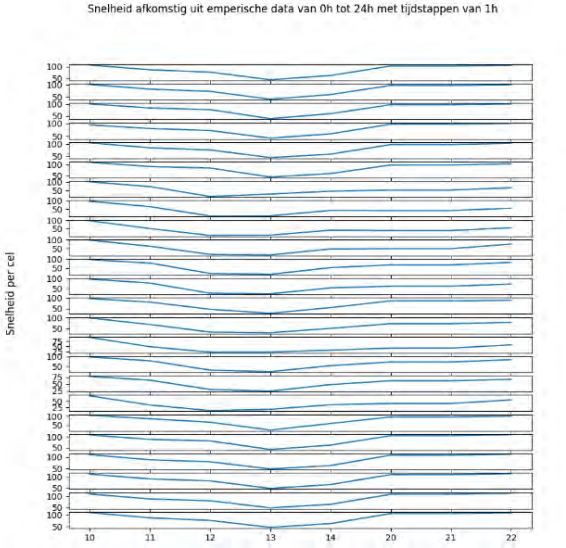
Figuur 52:

Visualisatie van Snelheid afkomstig van Data A1/A30



Figuur 53:

Visualisatie van Snelheid afkomstig van Data A1/A30



Casestudie 3: knooppunt Hoevelaken (A1/A28)

Inleiding

Knooppunt Hoevelaken kent door de A28, die de A1 op de stad Utrecht aansluit, regelmatig congestie. Daarnaast ligt het direct aan Amersfoort vast wat nadelen met zich meebrengt. Hierdoor zijn er al rigoureuze verbeteringen aan het systeem voorgesteld door Rijkswaterstaat. Voor de uitvoering moet landelijk besluit over de stikstofproblematiek worden afgewacht (Ministerie van Infrastructuur en Waterstaat, 2022). Bij goedkeuring volgen er drastische veranderingen aan het knooppunt. Dit is te zien in figuur 55.

Figuur 55:

Voorgesteld Knooppunt Hoevelaken

Noot. Ministerie van Infrastructuur en Waterstaat (2022)

Figuur 57:

Detectielussen Knooppunt Hoevelaken

Noot. NDW (2022)

Beschrijving

Knooppunt Hoevelaken is een klassiek klaverbladknooppunt (zie figuur 54 en 56). Het is een geschikte kruising aangezien het met enige regelmaat congestie kent en het daarnaast onafhankelijk is van andere verkeerssystemen. In dit onderzoek wordt de verkeersrichting van de A1 naar het noordwesten en de A28 naar het zuidwesten bestudeerd. Deze richtingen zijn gekozen omdat ze gemiddeld gezien meer congestie kennen. De snelwegen zijn voornamelijk tweebaanswegen. Binnen het klaverblad is er echter een 2x2 formatie. (Er is dus één tweebaansweg voor doorgaand verkeer en één tweebaansweg voor afslaand verkeer). Alleen de A28 in zuidelijke richting is een driebaansweg. De data voor de gewenste periode was in zijn geheel beschikbaar. De gebruikte detectielussen kenden geen storingen gedurende de opgevraagde periode (NDW, 2022).

Figuur 54:

Satellietfoto Knooppunt Hoevelaken

Noot. Aangepast overgenomen van Google (Z.D.)

Figuur 56:

Satellietfoto Knooppunt Hoevelaken

Noot. Aangepast overgenomen van Google (Z.D.)

Modellering

Dit model maakt gebruikt van 40 detectielussen. De complexiteit van dit model maakt het moeilijk en tevens onduidelijk om de plaatsing van cellen te beschrijven. Daarom is dit in de twee satellietfoto's gevisualiseerd (zie figuur 54 en figuur 56). De richting van de gemodelleerde snelwegen is telkens de richting die zich aan de bovenkant van de nummers bevindt. De relatief aparte nummering heeft te maken met de werking van het model. Het model output namelijk alleen naar cellen met een hoger nummer (value). Voor duidelijkheid wordt daarom voor elke reeks met een tiental gestart en gelden voor de laatste cellen honderdtallen. Alle volgorden van getallen moeten immers oplopend zijn. Het model kent 2 startcellen (1 en 10) en 2 eindcellen (101 en 206). Het heeft in totaal 8 op- en afritten. De lengtes van de cellen zijn zo gekozen dat eraan de CFL-conditie is voldaan (zie *Hoofdstuk 1; Bron 5* en *Hoofdstuk 4*). Voor Q, v_f, K_{cr} en K_{jam} is er respectievelijk gekozen voor 1,0; 1,0; 2,35; 4,0. Voorderest zijn er geen wijzigingen aan de data toegebracht. De waarde voor K_{cr} is zo gekozen dat het zich op een omslagpunt tussen condities met een opstopping en condities met beheersbare congestie bevindt.

Resultaten

In de resultaten is een ochtendspits herkenbaar. Dit is te verklaren omdat de onderzochte richting in de richting van de randstad is. Dit is de werkomgeving van veel Nederlanders en hier wordt daarom vooral 's ochtends op aangereden. Daarnaast ziet men dat de intensiteit op de A1 aanzienlijk lager is dan de intensiteit op de A12. Tevens is de A1 intensiteit op de A1 gedurende de hele dag redelijk gelijk verdeeld, terwijl de A12 een ochtend- en een avondspits kent. Hier is 's avonds bijvoorbeeld in cel 20 congestievorming (zie figuur 61). Daarnaast is 's ochtends duidelijk congestievorming te herkennen op de A12 (zie figuur 58). Ook is er een hoge intensiteit in cel 201 (zie figuur 58). Dit is de cel waar de A1 vanuit noordwestelijke richting bij de A12 voegt. Enkel kent het altijd dezelfde vorm in de grafiek wat laat zien dat het omstandigheden met een vrijestroomsnelheid zijn.

Het model levert het volgende resultaat:

$$TTS: \quad 12474849$$
$$N\_tot: \quad 68768$$
$$TTS/N\_tot: \ 181{,}4$$

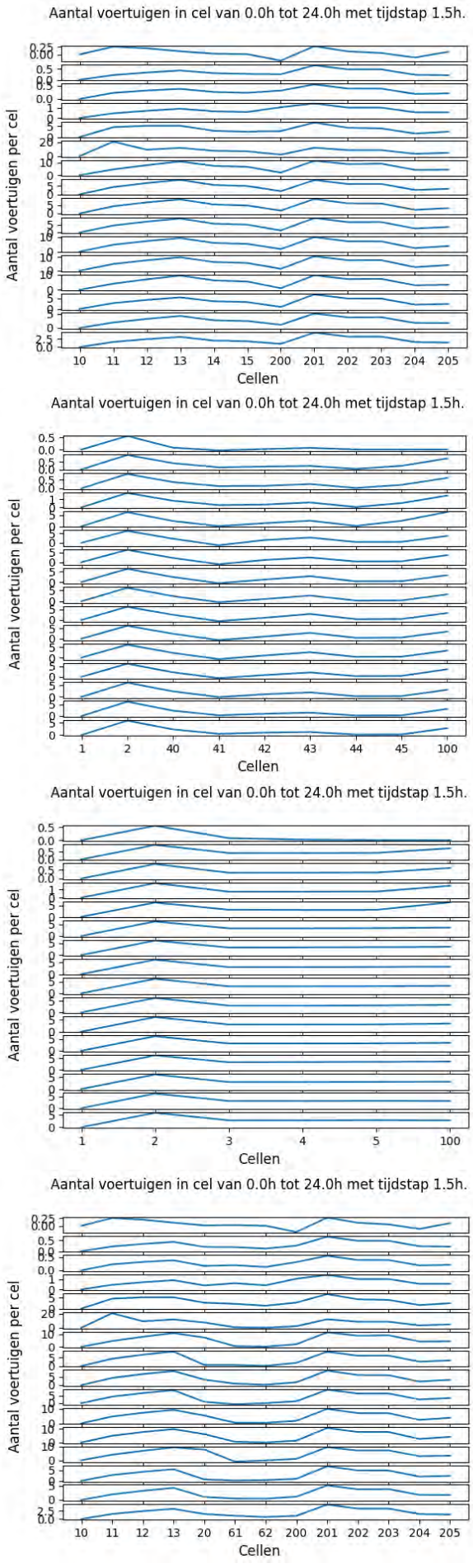
181,4 (s) lijkt hoog vergeleken met casestudie 1 en 2. Echter, er wordt bijvoorbeeld op de A12 al gekeken naar een totale lengte van 5,4 kilometer. Wanneer dit met een snelheid van circa 100 kilometer per uur wordt gereden doet men hier circa 3 minuten over. Dit verklaart dus de hoge waarde van de TTS/N_tot vergeleken met casestudie 1 en 2 die een kleiner en dus korter systeem beschrijven.

Optimalisering

Er is alleen congestievorming in cel 11. Dit wordt veroorzaakt door problemen met de inputcel (zie *hoofdstuk 6; Casestudie 2).* Hier valt niks aan te veranderen. Ook zijn er grote veranderingen in het aantal voertuigen in cel 20 te herkennen. Dit is echter een cel die door zijn ligging complex is. Het volgt immers uit cel 13 waarna het naar twee cellen output waarvan één ook weer een andere oprit kent. Het kent dus een inputcel die naar twee ontvangende cellen output en zelf output het ook naar twee ontvangende cellen. Lichte opbouw van congestie in deze cel is dus door samenloop van verschillende splitsingsverhoudingen te verklaren. Deze congestie propageert niet door het model en heeft dus ook geen grote gevolgen voor het knooppunt. Op schoonheidsfoutjes na loopt dit verkeerssysteem dus op een degelijke wijze en kan er geen optimalisering worden aangebracht.

Figuur 58, 59, 60, 58:

4 Visualisaties model A1/A28



Vergelijking van resultaten

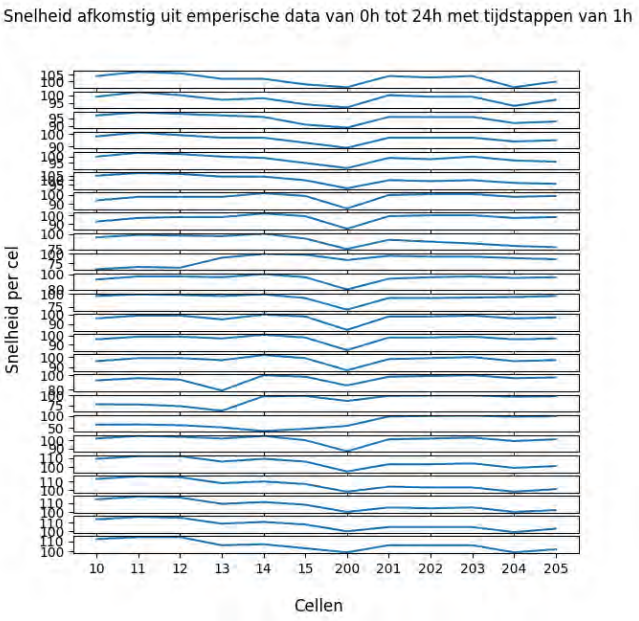
Een vergelijking van de snelheden verkregen in het model met de eigenlijke snelheden laat zien dat er een gemiddelde afwijking is van 13,79 (*km/h*). Aangezien het model geen rekening houdt met een mogelijke capacity drop of afname van homogeniteit worden (kleine) ochtendspitsen en avondspitsen niet zichtbaar, terwijl van deze in de werkelijkheid wel spraken zijn. In figuur 63 is te zien dat er in het model enkel van een spits spraken van 08:00 tot 09:00. De data, zoals te zien in figuur 62, kennen ook een avondspits die moeilijk herkenbaar is in het model. Men ziet enkel kleine veranderingen in cel 20.

Conclusie

Het model functioneert juist. Het model blijkt door zijn vereenvoudiging van de werkelijkheid dus sommige filesituaties wel op te pakken, terwijl het bij andere situaties in omstandigheden zonder congestie blijft verkeren. Er is daarom geen optimalisering aan het systeem mogelijk. De gemiddelde afwijking bedraagt 13,79 (*km/h*).

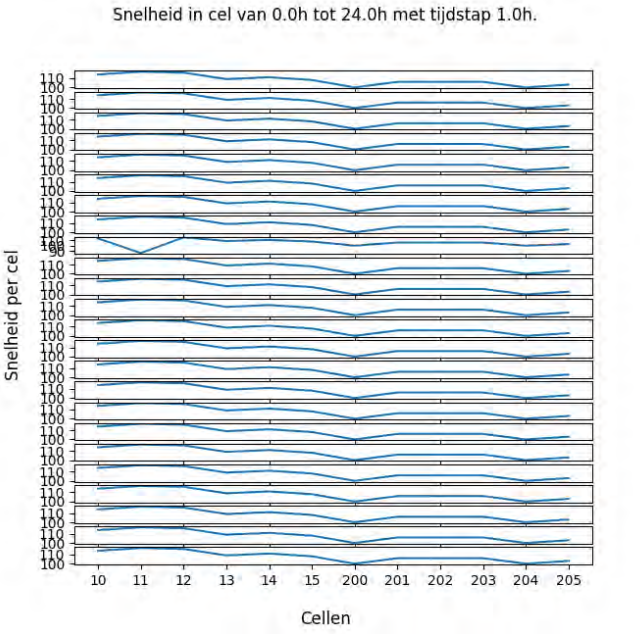
Figuur 62:

Visualisatie van Snelheid afkomstig van Data A1/A28



Figuur 63:

Visualisatie van Snelheid afkomstig van Model A1/A28



Casestudie 4: ring Eindhoven (A2/A50/A58)

Inleiding

De ring van Eindhoven is een van de ingewikkeldste verkeerssystemen die Nederland kent. Er is namelijk een extra N-weg die parallel loopt aan de A2 zodat de A2 zo weinig mogelijk op- en afritten kent. Dit betekent echter dat er op het breedste punt acht verschillende wegen naast elkaar liggen (zie figuur 64). Daarnaast kent het op korte afstand de oprit en afrit van twee verschillende snelwegen: de A50 en de A58 (zie figuur 65 en 67). De A58 kent regelmatig congestie doordat het de slagader tussen Eindhoven en Breda vormt. Op de A2 komt ook regelmatig file voor. Het vormt immers een ring om Eindhoven, de grootste stad van Noord-Brabant.

Beschrijving

Dit knooppunt valt niet te beschrijven in een simpele vakterm. Voor verduidelijking zijn satellietfoto's beschikbaar of moet de bron geraadpleegd worden (Google, Z.D.). De wegen zijn voornamelijk tweebaanswegen. Enkel verbreden ze vaak door opritten tot drie- of vierbaanswegen.

In dit model wordt gekeken naar de zuidelijke richting van de A2 en de richtingen van de A50 en A58 die invoegen in de A2. Dit is niet op werkelijke gegevens gebaseerd. Dit is het geval, omdat nu zo lang mogelijk naar de ring om Eindhoven zelf gekeken wordt waarbij ook de gevolgen van de A50 en de A58 zichtbaar zijn.

Daarnaast is dit knooppunt geschikt omdat het veel detectielussen bezit, het kent met enige regelmaat congestie en het is redelijk onafhankelijk. De gebruikte detectielussen kenden geen storingen gedurende de opgevraagde periode (NDW, 2022).

Figuur 64:

Satellietfoto Ring Eindhoven

Noot. Google (Z.D.)

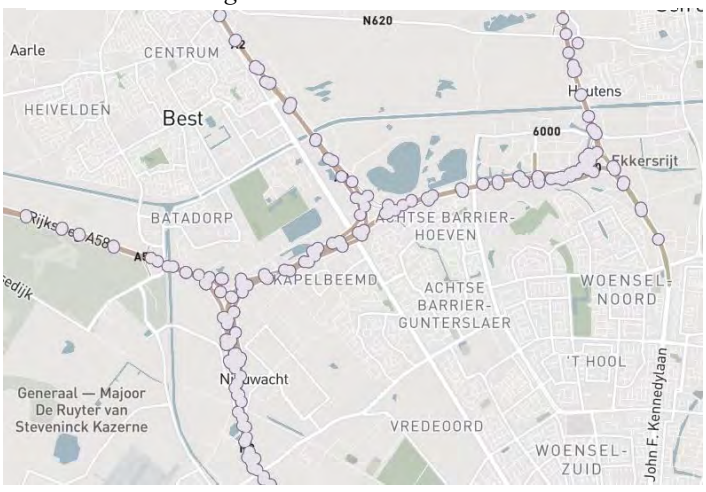
Figuur 65:

Satellietfoto Ring Eindhoven

Noot. Aangepast overgenomen van Google (Z.D.)

Figuur 66:

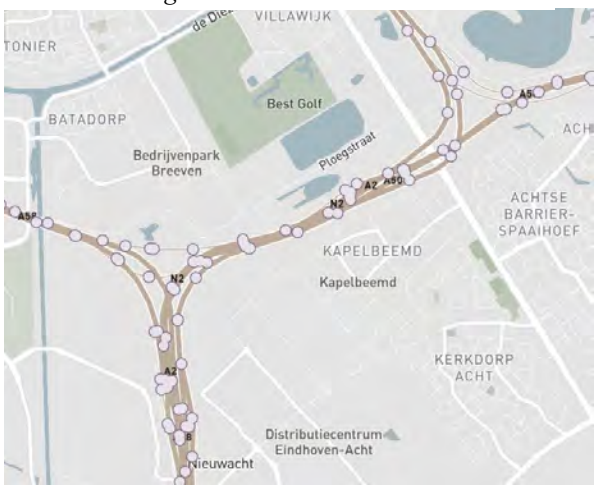
Detectielussen Ring Eindhoven



Noot. NDW (2022)

Figuur 67:

Detectielussen Ring Eindhoven



Noot. NDW (2022)

Figuur 68:

Satellietfoto Ring Eindhoven

Noot. Aangepast overgenomen van Google (Z.D.),

Modellering

Dit model maakt gebruik van 61 detectielussen. De plaatsingen van de cellen is op de satellietfoto's aangegeven. Figuur 65 kent de grote lijnen (d.w.z. de waarden van de snelwegen), terwijl figuur 68 specificaties geeft (d.w.z. de N-weg en op- en afritten). Het model kent 3 startcellen, cel 1, 11 en 50. Het kent 2 eindcellen, één van de A2 en één van de N2, respectievelijk cellen 144 en 112. Het model heeft 54 cellen en in totaal 7 op- en afritten. De lengtes van de cellen zijn zo gekozen dat eraan de CFL-conditie is voldaan (zie *Hoofdstuk 1; Bron 5* en *Hoofdstuk 4*).

Het is een complex model dat door zijn grote opzet redelijk met de werkelijkheid overeenkomt. Hierdoor is ervoor Q, v_f, K_{cr} en K_{jam} respectievelijk 1,0; 1,0; 3,3; 4,0 gekozen. Daarnaast is gekozen om q met een factor van 1,1 te verhogen. Hierdoor worden de dagen met meer congestie gemodelleerd. Dit zijn immers de dagen waarop men het verkeer wil verbeteren. Dit zorgt nu voor omstandigheden met congestie.

Resultaten

De resultaten zijn:

$$TTS: \quad 191959362$$
$$N\_tot: \quad 91159$$
$$TTS/N\_tot: 2106$$

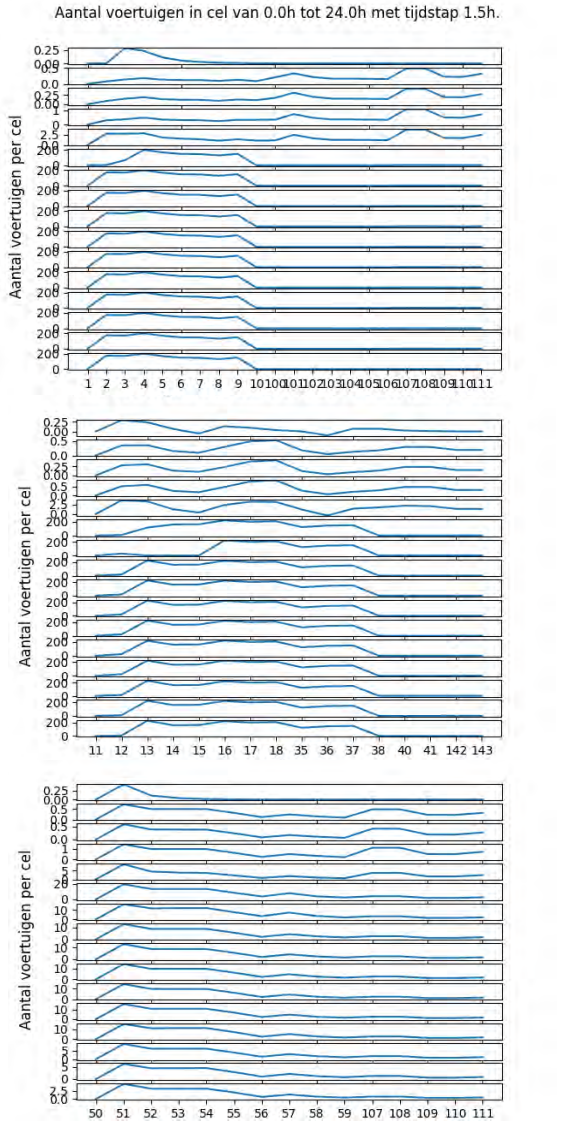
Figuur 69:

Satellietfoto Cel 37 (bovenste weg)

Noot. Google (Z.D.)

Figuur 70, 71, 72:

Visualisatie model A2/A50/A58 Zonder Aanpassingen



Het model kent dus veel congestie. Dit komt door twee opstoppingen in cel 9 en cel 37. Cel 37 kent een kruising waardoor er kort een vierbaansweg is (zie figuur 69). Hierdoor is Q in deze cel laag. Er is echter een groter probleem, een van de afritten is namelijk als oprit ingesteld. Dit zorgde in totaal voor een input van 15500 voertuigen in plaats van een output. Dit probleem deed zich ook nog in een andere cel voor. Beide zijn verholpen. Nieuwe resultaten laten echter nog steeds dezelfde problemen zien in cel 37 en 38. Al dan niet in mindere mate. Hierdoor is besloten op Q in cel 37 en 38 met een factor van 1,5 te verhogen, om de fout van de vierbaansweg te corrigeren. Daarnaast vertoont cel 12 problemen die ook zichtbaar zijn bij *Hoofdstuk 6; Casestudie 2*. Vanuit de startcel ontvangt het veel voertuigen, maar het is niet in staat om deze verder te verplaatsen. Hierdoor is besloten om Q van cel 13 met een factor 1,5 te verhogen. Samen levert dit volgende resultaat:

$$TTS: \quad 17134391$$
$$N\_tot: \quad 126621$$
$$TTS/N\_tot: \ 135{,}3$$

Toch is er nog steeds van een opstopping spraken op de A50. Deze wordt veroorzaakt door een terugkerende probleem omgaande de output uit de startcellen. Het valt niet te verhelpen. Daarom is besloten de intesiteit van de startcel van de A50, cel 1, met een factor van 0,91 te verlagen. Dit is de minimale factor waarbij er net geen opstopping ontstaat, maar wel congestie. Hierbij is het resultaat:

$$TTS: \quad 20799018$$
$$N\_tot: \quad 134306$$
$$TTS/N_tot: 154{,}9$$

Optimalisering

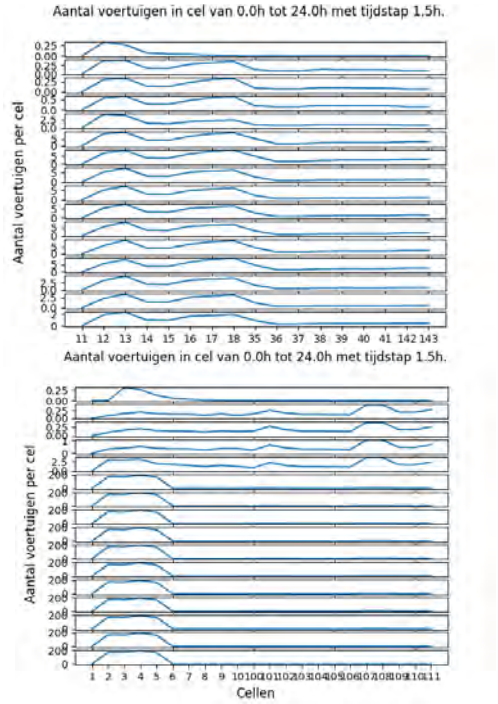
In het systeem vormt zich gedurende de ochtend- en avondspits congestie in de cellen 1 t/m 6 (zie figuur 75). Voorderest is er geen congestie in het model te herkennen. Doordat er geen spraken is van opritten vóór cel 6 kan men geen gebruik maken van RM. Er wordt daarom gekeken naar de implementatie van VSL. De snelheid in de cellen voorafgaand aan cel 6 is verminderd tijdens uren met hoge intensiteit. De snelheid is hierbij verlaagd tot 40 (km/h), de naleving is op 0,6 gesteld. Dit geldt vanaf 16:00 tot 19:00. Dit levert het volgende resultaat op:

$$TTS: \quad 21472424$$
$$N\_tot: \quad 134299$$
$$TTS/N\_tot: 159{,}9$$

In overeenkomst met casestudie 2 is te zien dat de tijd per reiziger toeneemt. In dit geval met $(-)3{,}13$ %. Enkel is er belangrijkere verbetering te zien in de figuur 78: de intensiteit is beter verdeeld over de cellen. Dit verhoogt de homogeniteit onder bestuurders (iets wat niet in het model verwerkt is). Dit heeft positieve gevolgen op de T_s, het aantal verkeersongelukken en de klimaatuitstoot (zie *Hoofdstuk 1; Bron 4*). Voorderest treedt er geen congestie op in het model gedurende de onderzochten periode. Er zijn dus andere geen optimalisering mogelijk.

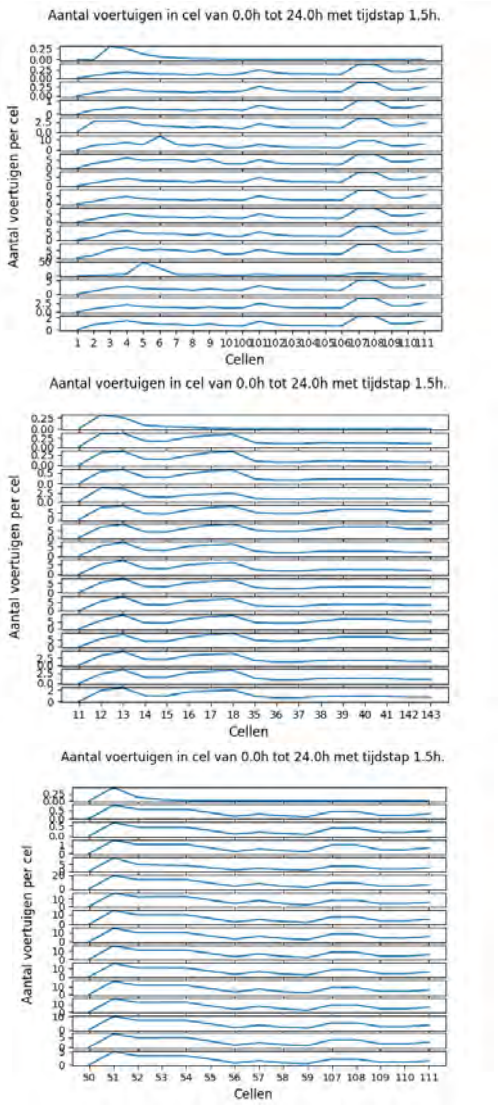
Figuur 74 en 74:

Visualisatie model A2/A50/A58 na Eerste Aanpassingen



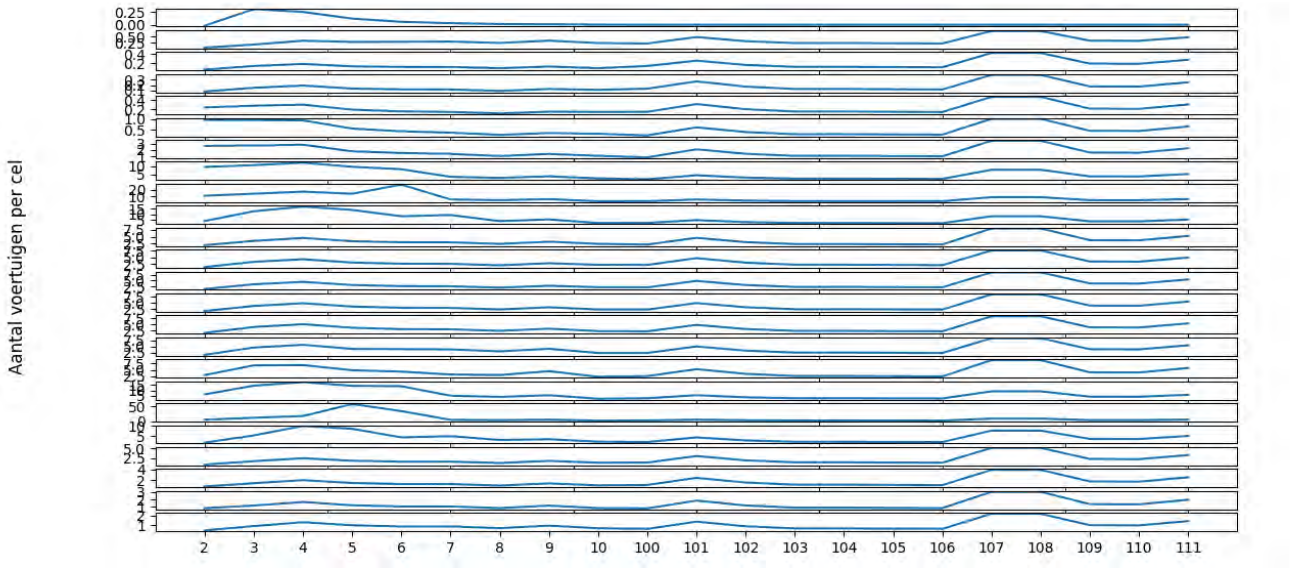
Figuur 75, 76, 77:

Visualisatie model A2/A50/A58 na Tweede Aanpassingen



Figuur 77:

Visualisatie model A2/A50/A58 Met VSL



Vergelijking van resultaten

Een vergelijking van de snelheden verkregen in het model met de gemeten snelheden laat zien dat er een gemiddelde afwijking is van 12,75 (*km/h*). Dit is een vergelijking met de simulatie waarbij VSL niet actief zijn. De afwijking is vergelijkbaar met eerdere resultaten. Er is echter een groot verschil wanneer men naar de cellen 50 t/m 59 kijkt: 18,43 (*km/h*). Andere gebieden, d.w.z. cel 1 t/m cel 111 en cel 11 t/m 143, kennen een klein verschil, respectievelijk 14,59 (*km/h*) en 8,62 (*km/h*).

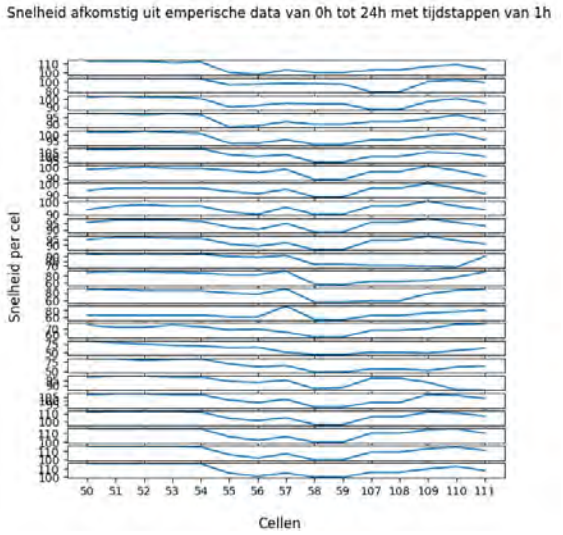
Conclusie

De grote afwijking van snelheden in cellen 50 t/m 59 is te verklaren door het feit er hier congestie onopgemerkt voorbijgaat. Er wordt gebruik gemaakt van een inputcel die ook onder omstandigheden met congestie input naar het systeem. Dit is relatief weinig door de paradox van de intensiteit (zie *Hoofdstuk 1; Bron 2*). Hierdoor kan het systeem dit wel aan als de congestie eerder in het systeem veroorzaakt is (d.w.z. een deel dat niet gerepresenteerd is in het model). Aan één van de eisen die verplicht is gesteld voor het kiezen van een onderzoeksgebied is dus niet voldaan: het feit dat een onderzoeksgebied zo onafhankelijk mogelijk moet zijn (zie *Hoofdstuk 2*). Dit maakt het dat model in deze rij van cellen geen juiste resultaten geeft. De enige oplossing voor dit probleem is om nog meer cellen aan het systeem toe te voegen, zodat het systeem waar de congestie veroorzaakt wordt, wel geïncorporeerd is in het model. Dan moeten er echter meerdere verkeersknooppunten met elkaar verbonden worden wat buiten de scope vaan dit onderzoek gaat. Deze fout is dus geen fout van het model. Er is enkel een fout gemaakt tijdens het bepalen van het onderzoeksgebied en later het selecteren van de data.

Ten conclusie, het model functioneert naar behoren. Echter, congestie gaat onopgemerkt voorbij door onjuiste keuze van het onderzoeksgebied. Er is in de realiteit immers een avondspits (zie figuur 79). Door de structuur van het systeem is er naast VSL, wat de T_s met 3,13 % verhoogde, geen optimalisering mogelijk.

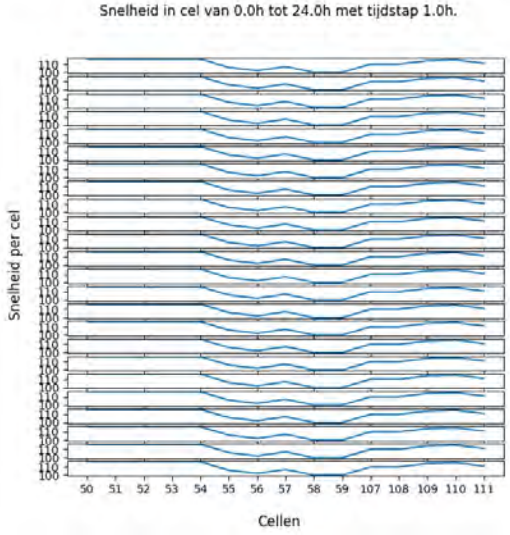
Figuur 79:

Visualisatie van Snelheid afkomstig van Data A2/A50/A58



Figuur 78:

Visualisatie van Snelheid afkomstig van Model A2/A50/A58



Hoofdstuk 7: herhalingsexperiment

Inleiding

Aangezien casestudie 2 het bruikbaarst is voor optimalisatie is er een herhalingsexperiment uitgevoerd gedurende een andere periode. Dit bevestigt voor de laatste keer de werking van het model. Daarnaast toont het de juistheid van de gebruikte data aan. In deze herhalingsexperiment kijkt men wederom naar een periode van vijf werkdagen. Enkel vallen deze nu in de periode van 13 t/m 17 september 2021. Dit is twee weken na het einde van de zomervakantie en valt in een periode buiten coronamaatregelen (RIVM, 2022).

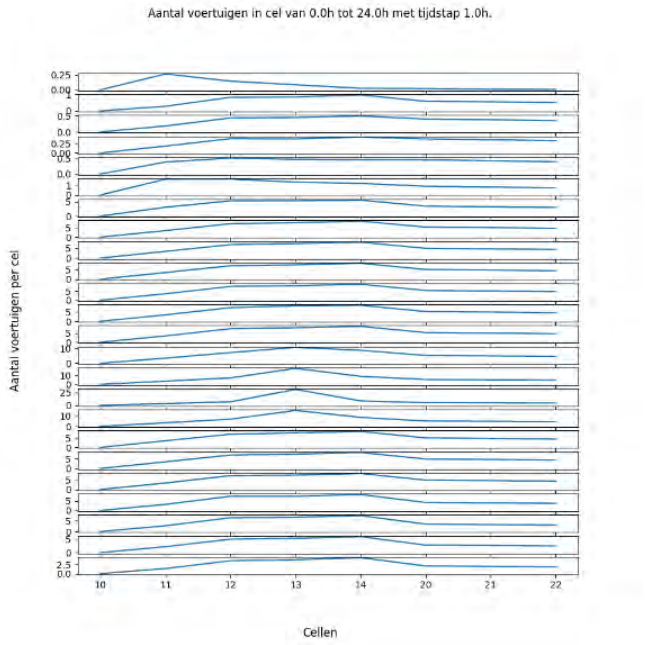
De data voor de gewenste periode was in zijn geheel beschikbaar. De gebruikte detectielussen kenden geen storingen gedurende de opgevraagde periode (NDW, 2022). Voor een inleiding en beschrijving van het systeem moet *Hoofdstuk 6; Casestudie 2* geraadpleegd worden.

Modellering

Het systeem is identiek aan degene uit casestudie 2. Voor de gebruikte verhoudingen van Q, v_f, K_{cr} en K_{jam} ten opzichte van de empirische data gelden respectievelijk 1,2; 1,0; 2,3 en 4,0. Dit zijn exact dezelfde verhoudingen als de gebruikte in casestudie 2. Wederom is q met een factor van 1,75 vermenigvuldigd om de complexiteit van het systeem door onder andere het kruispunt te midden van het knooppunt te representeren. Enkel is Q van cel 11 verlaagt tot 856,6 (v/h). Zo wordt een opstopping voorkomen, maar blijft zo veel mogelijk congestie in stand. Dit wordt veroorzaakt door het terugkerend probleem waarbij een startcel te veel output naar de volgende cel (zie *hoofdstuk 6; Casestudie 2*). Deze aanpassingen is als enige niet in overeenkomst met casestudie 2. Hier een waarde van 932,9 (v/h) gebruikt. Dit is vergelijkbaar, maar het toont alvast dat het huidige herhalingsexperiment meer congestie met zich meebrengt omdat het een lagere waarde vereist.

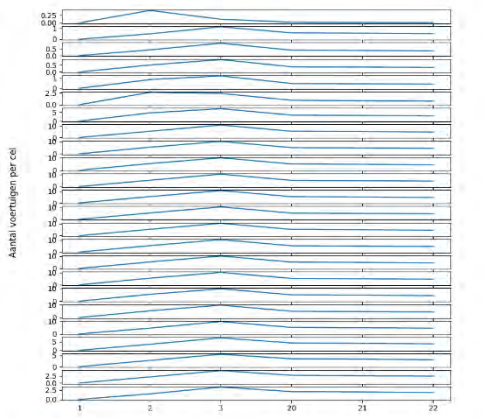
Figuur 81:

Visualisatie model Herhalingsexperiment A1/A30



Figuur 80:

Visualisatie model Herhalingsexperiment A1/A30



Resultaten

Het resultaat van het herhalingexperiment is:

$$TTS: \quad 4980099$$
$$N\_tot: \quad 796338$$
$$TTS/N\_tot: \ 62{,}5$$

In casestudie 2 bedroeg de gemiddelde bestede tijd per reiziger 59,3 (s). Er is dus een verschil van 3,2 (s) op ongeveer één minuut. Zoals al bij *Hoofdstuk 7; Modellering* voorspelt is, kent deze periode net wat meer congestie, dit is terug te zien in deze hogere waarde. Voor de tijd per reiziger. De overeenkomst met casestudie 2 is ook terug te zien in de grafieken (zie figuur 81 en 82), op hetzelfde punt (d.w.z. cel 13) bouwt congestie op. Verschillende in hoeveelheid file worden verklaard door verschillende factoren waar geen duidelijke uitspraak over te doen is. Mogelijk heeft het met het jaargetijden of een periode met minder corona te maken. Wat ook opvalt, is dat de piek in het herhalingsexperiment pas twee uur later zichtbaar is.

Optimalisering

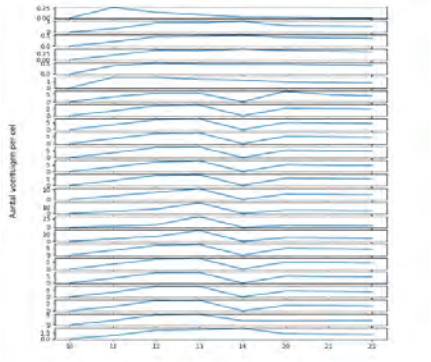
Ramp metering

Wanneer men gedurende dezelfde periode (d.w.z. 06:00 tot 22:00) RM implementeert op dezelfde cel (d.w.z. cel 14) geeft dit het volgende resultaat:

$$TTS: \quad 4501645$$
$$N\_tot: \quad 79633$$
$$TTS/N\_tot: \ 56{,}5$$

Figuur 82:

Visualisatie model Herhalingsexperiment A1/A30 met RM



Hier wordt een procentuele verandering behaald van $(-)9{,}6$ %. Dit is vergelijkbaar met de eerder behaalde $(-)10{,}6\%$. Dit bevestigt de werking van RM.

Variabele snelheidslimieten

Ook wanneer VSL op dezelfde manier worden geïmplementeerd ziet men hetzelfde resultaat (d.w.z. een snelheid van 40 (km/h) in cellen 10 t/m 14 van 06:00 tot 22:00):

$$TTS: \quad 5223876$$
$$N\_tot: \quad 79631$$
$$TTS/N\_tot: \ 65{,}6$$

Hier veranderd de tijd per reiziger procentueel met 4,96 %. Dit is vergelijkbaar met casestudie 2 waar de tijd per reiziger met 6,91 % toeneemt. Het is enkel circa 2 procentpunt lager. Dit is verklaren door het feit dat deze periode een hogere mate aan congestie kent. Het implementeren van een lagere snelheid heeft dan een minder groot effect op de al laag liggende snelheid. In de bijbehorende grafiek is wederom een hogere mate aan homogeniteit te zien (zie figuur 85).

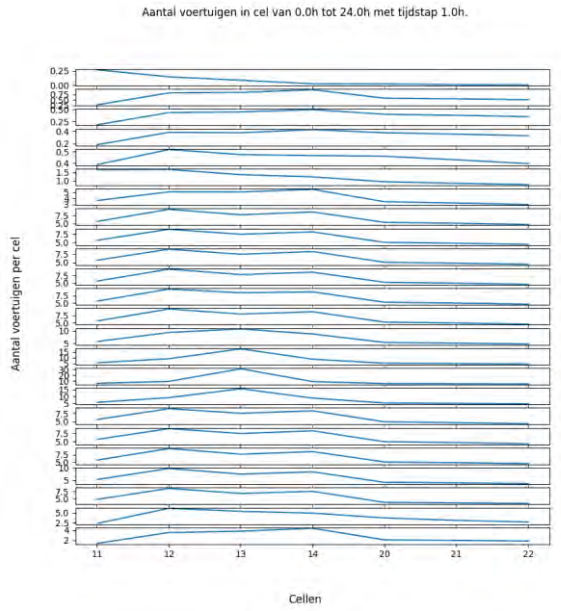
Langetermijnverbetering

Ten slotte is er naar langetermijnverbeteringen gekeken. Cel 10 t/m 14 zijn aangepast tot $v_f = 110$ en $\lambda = 2$. Dit is een ruime schatting van de mogelijke implementatie van een fly-over. Het geeft de volgende resultaten:

$$TTS: \quad 3955863$$
$$N\_tot: \quad 79633$$
$$TTS/N\_tot: \ 49{,}7$$

Figuur 84:

Visualisatie model Herhalingsexperiment A1/A30 met VSL



Dit is een procentuele verandering van $(-)16{,}2$ %. Dit is significant lager dan de eerder behaalde resultaten van 21,4 %. Wanneer men de hoeveelheid voertuigen op een bepaald moment op A1 tussen de twee studies vergelijkt, valt het op dat de verhoudingen $A1:A30$ in het herhalingsexperiment meer naar de A1 ligt. Hierdoor heeft veranderen van de situatie op de A30 een minder grote invloed. De hogere dichtheid op de A1 is niet te verklaren door een verschil in inputverhoudingen in de casestudie en het herhalingsexperiment. Deze liggen respectievelijk op 0,52 en 0,51 (gekeken vanaf de A30). Dit verschil is dus veroorzaakt door externe factoren, zoals jaargetij.

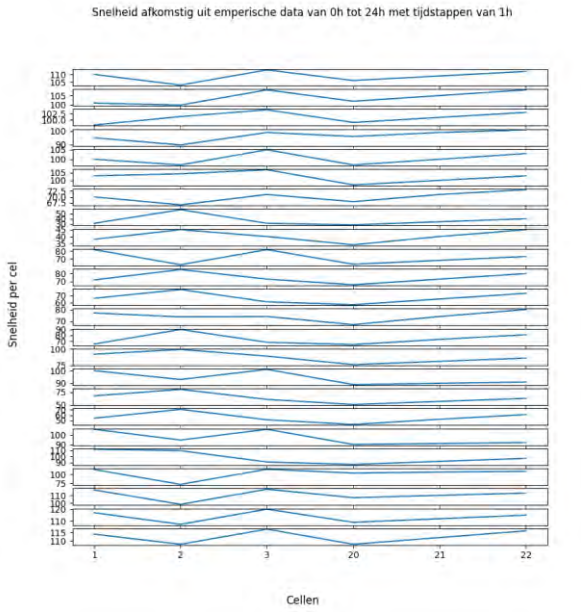
Vergelijken van resultaten

Nog steeds kent dit systeem in werkelijkheid meer congestie dan dat zichtbaar is in het model. Dit wordt veroorzaakt door de plaatsing van de startcellen. Dit is een terugkerend probleem (zie *Hoofdstuk 6; Casestudie 2*). Het verschil in snelheden tussen de data en het model bedraagt 22,6 (km/h). Afgezien van het feit dat dit geen positief resultaat is, is het wel in overeenstemming met het verschil gemeten in casestudie 2. Dat dit verschil in de herhalingsexperiment iets lager ligt, komt door de hogere mate aan congestie.

De snelheidsdata van deze periode is in cellen 1 t/m 3 opvallend (zie figuur 85): de pieken wisselen per uur af tussen de verschillende cellen. Dit lijkt op een harmonica-effect van shockwaves die zich door het systeem verplaatsen (zie *Hoofdstuk 2; Bron* 2). Door de sterke vereenvoudigingen van het model is dit niet terug te zien.

Figuur 83:

Visualisatie van Snelheid afkomstig van Data A1/A30



Conclusie

Het model uit de casestudie en het herhalingsexperiment komt overeen. Daarnaast komen alle waarde, verbeteringen en andere testresultaten overeen. Kleine verschillen tussen de twee zijn telkens verklaarbaar. Hieruit valt de conclusie te trekken dat het model een juist beeld levert dat overeenkomt tussen verschillende periode. Al is het vereenvoudigd.

Hoofdstuk 8: conclusie

In dit onderzoek is onderzocht *'hoe de filedrukte verminderd kan worden, door middel van een voor iedereen begrijpelijk wiskundig model?'*. Hiervoor is een onderzoek uitgevoerd naar hoe snelwegen gemodelleerd worden. Vervolgens is een model opgesteld waarmee via vier casestudies resultaten verkregen.

Voorafgaand aan de casestudies, is uit de resultaten van vier testexperimenten gebleken dat de theoretische en computationele werking van grafentheorie juist is toegepast. Computationele resultaten komen immers overeen met achterliggende gegevens en wiskundige berekeningen. Tevens zijn binnen het wiskundig model onderdelen van het CTM en het FD juist vervaardigd. RM behaalt een verbeteringen van de T_s van 2% en 288% (veroorzaakt door het oplossen van een opstopping). Het is op een juiste wijze in het model toegepast. Dit geldt ook voor andere optimaliserende algoritmen als VSL.

Dit onderzoekt bekijkt vier (Nederlandse) snelwegknooppunten met complexiteiten van klassieke knooppunten tot een van de moeilijkste van Nederland. Onderzoek naar deze vier casestudies toont een gemiddelde snelheidsafwijking van $15{,}23\ (km/h)$. Kenmerken van verkeerssystemen zijn echter juist terug te zien in de resultaten. Kortom, er is een wiskundig model gepresenteerd dat, al vereenvoudigd, in een hoge mate overeenkomt met de werkelijkheid.

Tevens wordt filedrukte verminderd door het toepassen van intelligente verkeersstrategieën. Naast een kwalitatieve onderbouwing, neemt de toepassing in casestudies kwantitatieve resultaten met zich mee. RM levert op de aansluiting Barneveld een verbetering van $10{,}6\ \%$ op (zie *Casestudie 2*). Voorderest verslechterd de situatie ten gevolgen van VSL met gemiddeld $5{,}02\ \%$ (genomen over casestudie 2 en 4). Bijbehorende grafieken worden echter gekenmerkt door hogere homogeniteit. Volgens de literatuur gaan hier voordelen voor de T_s mee gepaard (Hegyi et al, 2005). Over de precieze gevolgen zijn geen concrete uitspraken te doen. De gevolgen van homogeniteit zijn immers niet geïncorporeerd in het model. Daarnaast laten langetermijnverbeteringen op het knooppunt A1 en A30 een verbetering van 21,4% zien. Dit is in overeenkomst met onderzoek van grote instanties (MIRT, Gemeente Barneveld, 2018). Dit toont dat dit model, al vereenvoudigd, in staat is grote veranderingen aan verkeerssituaties correct te modelleren

Ten slotte laten de resultaten van het herhalingsexperiment, uitgevoerd naar casestudie 2 gedurende een andere periode, zien dat het model bestendig is tegen de tijd. Met nagenoeg gelijke parameters worden namelijk resultaten verkregen die minder dan een paar procentpunten verschillen. Indien dit niet het geval is zijn verschillen eenvoudig te verklaren.

Met dit onderzoek is aangetoond dat men snelwegen, met al hun nuances, correct in een simpel en begrijpelijk wiskundig model gebaseerd op grafentheorie kan verwerken. Waarnaast intelligente verkeersmanagementstrategieën kunnen worden toegepast. Deze strategie, waarvan de belangrijkste RM en VSL, hebben positieve gevolgen op verkeerssystemen in het model. Zij moeten vaker op snelwegen worden toegepast om filedrukte te verminderen. Tevens introduceert deze studie een wiskundig model die ieder in staat stelt elke (Nederlands) snelwegsysteem vereenvoudigd te modelleren om zo filevorming beter te begrijpen.

Hoofdstuk 9: discussie

Betrouwbaarheid en validiteit

Voor dit onderzoek is een wiskundig model geïntroduceerd om verkeerssystemen te modelleren en vervolgens filedrukte te verminderen. Dit model is gebaseerd op concepten uit *Intro to Traffic Flow Modeling and Intelligent Transport Systems* (École Polytechnique Fédérale de Lausanne, 2022) en verschillende wetenschappelijke artikelen waarnaar deze bron verwees. De correctheid van gebruikte algoritmen, in het in dit onderzoek geïntroduceerde model, is aangetoond in de bronnen waarnaar verwezen is. Hierdoor is de validiteit van het meetinstrument hoog. Daarnaast wordt de validiteit verhoogd doordat het model succesvol is toegepast op vier testexperimenten. Deze zijn allesomvattend voor aangetroffen verkeerssituaties in het eigenlijke onderzoek. Doordat resultaten van de testexperimenten overeenkomen met de meetresultaten van het eigenlijke onderzoek wordt de criteriumvaliditeit gewaarborgd. Het introduceren van extra algoritmen heeft ervoor gezorgd dat op- en afritten en intelligente verkeersstrategieën zijn gerepresenteerd in het model. Dit heeft de inhoudsvaliditeit verhoogd. Meetresultaten uit dit onderzoek komen overeen met resultaten van Rijkswaterstaat. Dit is te zien in bijvoorbeeld casestudie 2 (zie *Hoofdstuk 6; Casestudie 2).* Hierdoor wordt de begripsvaliditeit behouden. Aan de andere kant wordt deze verlaagt doordat in alle casestudies snelheidsverschillen tussen de data en het model van boven de $10\ (km/h)$ worden waargenomen. Doordat het model zijn parameters en input baseert op empirische data is de ecologische validiteit bewaard. Daarentegen moesten hieraan verhoudingsgewijs aanpassing gedaan worden waardoor niet met zekerheid gezegd kan worden of het in overeenstemming is met de werkelijkheid. Doordat er vier verschillende casestudies zijn uitgevoerd mag men spreken van onderzoekstriangulatie waardoor de interne validiteit hoog blijft. Er wordt hierdoor duidelijk dat causale verbanden in de resultaten niet door andere factoren beïnvloed worden. Daarnaast liet het herhalingsexperiment zien dat het model, onder invloed van dezelfde parameters, voor verschillende periodes juiste resultaten met zich meebrengt. Hierdoor mag men met zekerheid zeggen dat resultaten gegeneraliseerd mogen worden. Tevens laten de vier verschillende casestudies zien dat het model zelf generaliseerd mag worden en dus werkt voor elk snelwegsysteem dat vergelijkbaar is met een in Nederland.

Doordat er geen wijzigingen aan het model zijn aangebracht tijdens het onderzoek, is men ervan verzekerd dat een herhalingsexperiment dezelfde resultaten oplevert (indien de parameters gelijk blijven). Doordat elke casestudie door een universeel model gestandaardiseerd is, kent dit onderzoek een hoge betrouwbaarheid en zijn de resultaten van dit onderzoek valide.

Verwachtingen en resultaten

Het in dit onderzoek geïntroduceerde model is in staat snelwegsystemen succesvol te modelleren. Dit is in overeenkomst met de verwachtingen die uitspraken dat vereenvoudigde, maar realistische resultaten verkregen zouden worden. Dit model zou geen vernieuwende resultaten geven. In andere woorden: de resultaten zouden, al vereenvoudigd, in overeenstemming zijn met onderzoek van bedrijven of overheidsinstanties. Dit alleen door een nieuwe methodologie die vereenvoudigd is ten opzichte van bedrijven, overheidsinstanties en universiteiten. Deze vereenvoudigingen leiden tot minder vergelijkbare en minder realistische resultaten. Dit is duidelijk in de resultaten terug te zien. Veel van de casestudies laten een beeld van de werkelijkheid zien dat correct is, maar enkele belangrijke elementen mist. De gelijkenis met Rijkswaterstaat en onderzoekscentra is terug te zien in bijvoorbeeld casestudie 2. Langetermijnstrategieën zijn hierbij in overeenstemming met onderzoek van Rijkswaterstaat (MIRT, Gemeente Barneveld, 2018). Daarnaast wordt RM hier juist geïmplementeerd. Dit is in overeenstemming met het feit dat RM hier daadwerkelijk is toegepast en het theoretische kader waar RM toegepast is op een casestudie naar de A10 waar het een verbetering van circa 30 % bewerkstelligd (zie *Hoofdstuk 2; Bron 3*). Echter, VSL is het onderzoek niet duidelijk terug te zien omdat de gevolgen van homogeniteit onder bestuurders, door vereenvoudigingen, niet zijn geïncorporeerd in het model. Toch is de werkingen van VSL niet uitgesloten. Men zag in de grafieken immers duidelijk de mogelijk positieve effecten van VSL op de homogeniteit onder bestuurders. Tevens zijn complexere systemen als stoplichten niet geïntroduceerd in het model. Hierdoor moest, in casestudie 2, de totale intensiteit vanuit de startcellen verhoogd worden om voor de gevolgen van vereenvoudiging te compenseren (zie *Hoofdstuk 6; Casestudie 2)*.

Verklaringen voor afwijkingen

Verschillende resultaten zijn niet in overeenstemming met de verwachtingen. Ten eerste valt op dat systemen niet altijd congestie kennen, terwijl dit in de realiteit wel het geval is. Dit ziet men bijvoorbeeld in casestudie 4 in cel 50 t/m 59 (zie *Hoofdstuk 6; Casestudie 4*). In deze casestudie, maar ook het gehele onderzoek, zijn de startcellen zo gekozen dat vanaf deze cellen tot het knooppunt geen andere ingewikkelde systemen aanwezig zijn. Dit bedraagt vaak meerdere kilometers. Er was echter niet verwacht dat congestie tot zo ver in het systeem zou propageren. Hierdoor werd de paradox van de intensiteit overschreden (zie *Hoofdstuk 1; Bron 2*). Dit probleem is alleen te verhelpen door het uitbreiden van het systeem. Hierdoor kennen casestudies meerdere knooppunten, wat buiten de scope van dit onderzoek gaat. Het is dus bijna of niet te voorkomen, wil men de complexiteit van de casestudies handhaven. Immers, voor de correctheid van het model moet er een punt op het Pareto-frontier van correctheid en complexiteit gekozen worden (zie *Hoofdstuk 1; Bron 1*). De lezer moet beseffen dat deze studie gericht is op het geven van een zo begrijpelijk mogelijk beeld over verkeerstheorie op vwo zes niveau. Er wordt daarom voor een lage complexiteit en daarmee lage correctheid gekozen.

Ten tweede komen er regelmatig opstoppingen voor die niet uit zichzelf verholpen worden. Wanneer er te veel voertuigen in een cel aanwezig zijn neemt de snelheid af tot nul. Dit volgt uit het FD (zie *Hoofdstuk 1; Bron 5*). Deze opstoppingen zijn niet realistisch. De keuze voor complexere functies die dit voorkomen is echter in strijd met het doel dit onderzoek.

Ten derde is er onvoorziene congestievormingen in meerdere cellen. Dit is verholpen door het identificeren van afwijking in de maximale intensiteit (Q). Dit is alleen te verklaren door afwijkingen van de data. Dit is handmatig veranderd en heeft hiermee de validiteit van het onderzoek verminderd. Echter, er was niet verwacht dat data van een gerenommeerde bron als NDW afwijkt. Alleen door de hoeveelheid gebruikte detectielussen is de kans dat er één of meer afwijken of defect zijn groot. Het gaat buiten de scope van deze studie om dit te onderzoeken en mogelijk te verhelpen.

Ten slotte zijn er problemen met de startcellen. Startcellen proberen aan hun output te voldoen waardoor cellen verderop in het systeem overladen worden (zie *Hoofdstuk 6; Casestudie 2*). Dit is onrealistisch. In de realiteit zou er in de startcel immers ook congestie optreden Dit probleem is onverwacht, maar alleen te verhelpen door de complexiteit te verhogen.

Samenhang met theoretisch kader, de vernieuwing en de gevolgen van het onderzoek

Deze studie combineert alle concepten die zijn voorgesteld in het theoretisch kader. Het is hiermee een overkoepeld onderzoek over grafentheorie en verkeerstheorie dat helderheid biedt over de werking van en relatie tussen de twee vakgebieden. De vernieuwing vindt men vooral in de methodiek. Er is immers een nieuw universeel model geïntroduceerd. Het is een eenvoudig model dat toch in staat is om verkeerssituaties te modelleren en kent het de voornaamste verkeersstrategieën. Daarom draagt het model voor een individu bij aan het begrip van grafen- of verkeerstheorie. Modellen van grote bedrijven begrijpen is immers door specialisatie nagenoeg onmogelijk. Deze begrijpbaarheid is daarom de belangrijkste vernieuwing van dit onderzoek. Men is in staat om bij eigen gebruik van het model met verschillende verkeerssituatie te experimenteren en concepten zelf na te bootsen en te visualiseren. Hierdoor komt een gemiddelde bestuurder meer te weten over het ontstaan, de werking en de gevolgen van congestie. Dit leidt er tot dat bestuurders bewustere keuzen maken in situaties met congestie. Een desbetreffende verhoging van het bewustzijn heeft mogelijk positieve gevolgen op de jaarfilezwaarte. Men ziet immers bij VSL dat vaak rekening gehouden wordt met een lage naleving (Frejo, J. R. et al, 2018). Een hoge naleving voor alleen al VSL heeft positieve gevolgen. Dit is iets waar complexe(re) modellen niks aan bijdragen.

Verkeersmodellen zijn gevoelig en kleine veranderingen hebben dus grote gevolgen. Daarnaast hangt het verkeersmodel af van de zwakste schakel. Echter, de casestudies liegen niet over hun verbeteringen. Zo liet casestudie 2 serieuze verbeteringen zien en waren de effecten van VSL ook in andere casestudies zichtbaar. Dit onderzoek draagt dus bij aan het verminderen van filedrukte op

snelwegen. Zonder dit model, maar vooral zonder wiskundige modellen grote bedrijven, hadden snelwegen er vandaag de dag anders uitzien. Men is in het optimale geval compleet overgestapt op openbaar vervoer, maar waarschijnlijk kennen snelwegen een hogere mate aan congestie. Een ochtendspitsje is hier niks bij. Daarnaast dragen wiskundige modellen sterkt bij aan het voorspellen van snelwegstructuren zoals dit, al vereenvoudigd, is gedaan in *Casestudie 2*. Hierdoor wordt efficiënt de aanleg van nieuwe (snel)wegen bepaald. Dit heeft positieve effecten op de tijd en de portemonnee van de Nederlander.

Aanbevelingen

Uit deze studie volgt dat intelligente verkeersstrategieën in hogere mate moeten worden toegepast op Nederlandse snelwegen. Daarnaast toont dit onderzoek dat het door middel van een eenvoudig wiskundig model mogelijk is om de bestuurder meer te betrekken bij het besluitvormingsproces voor bepaalde verkeersstrategieën. Het in dit onderzoek geïntroduceerde model vormt hier een ideaal startpunt voor. Daardoor beveelt deze studie het in dit onderzoek geïntroduceerde model in zijn algemeenheid aan.

In tegenstelling tot modellen van Rijkswaterstaat of andere instituten, is het in dit onderzoek geïntroduceerde model sterk vereenvoudigd. Het houdt zich namelijk sterk aan de abstracte theorieën van de verkeerstheorie die zijn geïntroduceerd in het literatuuronderzoek. Als gevolg is het model gemakkelijker te begrijpen en uit te leggen, omdat het geen overbodigheden kent. Dit maakt het een perfect startpunt voor een bestuurder om meer te weten te komen over de werking van file waar een bestuurder dagelijks mee te maken heeft.

Beperkingen

Het in deze studie uitgevoerde onderzoek kent beperkingen. Ten eerste maakt men gebruik van data gegeven in uren. Voor nauwkeuriger resultaat moet er naar specifiekere data gekeken worden. Hiervoor moet de samenwerking met bedrijven of onderzoekscentra aangegaan worden. Er zijn namelijk geen open databronnen toereikender dan NDW. Ten tweede zijn er beperkingen op het gebied van modelcomplexiteit waardoor het model niet in staat is om de werkelijkheid perfect te representeren. In het model wordt geen rekening gehouden met:

- Microscopische keuzen (d.w.z. keuzen van specifieke voertuigen. Er wordt nu gebruik gemaakt van een macroscopisch model).
- Complexere verkeerssystemen als stoplichten of rotondes.
- Anticipatie van bestuurders op congestie verder in een systeem (denk hierbij aan het nemen van een andere route of het aanpassen van de vertrektijd).
- Gevolgen van homogeniteit onder bestuurders.
- Gevolgen van verkeerssituatie of incidenten (denk hierbij aan het af afsluiting van rijbanen of de gehele weg, het omleiden van verkeer of implementeren van verkeersstrategieën in zijn algemeenheid).
- Weersomstandigheden (denk hierbij aan de gevolgen van bijvoorbeeld sneeuwval).
- Vorm van wegbezetting (d.w.z. verhouding tussen vrachtverkeer, personenverkeer voor werk of privé. Hierbij moet onder andere de voertuiggrootte in acht genomen worden).

Alle beperkingen zijn te verhelpen door het verhogen van de complexiteit van het model (d.w.z. het toevoegen van nieuwe functies aan het model). Het gaat echter voorbij aan het doel van deze studie om deze beperkingen te onderzoeken. Ten derde biedt dit onderzoek door praktische problemen (d.w.z. de computerkracht) geen volledig inzicht in verkeersmodellering. Optimaliseerde algoritme konden namelijk niet in hun volledigheid benut worden. Ten vierde kijkt het model alleen naar de gevolgen voor de totaal bestede tijd en tijd per reiziger. Factoren als de uitstoot van broeikasgassen en geluidsoverlast zijn niet vertegenwoordigd in het model. Dit gaat echter voorbij aan de scope van deze studie.

Hierdoor is het voorgestelde model niet in staat uitspraak te doen over een groot aantal verkeerssituatie. Het model wijkt in zulke situaties namelijk af.

Vervolgonderzoek

Door het groot aantal beperkingen kent vervolgonderzoek uiteenlopende mogelijkheden. Hedendaags universitair onderzoek focust zich op het implementeren en combineren van microscopische met macroscopische modellen. Dit geeft immers het accuraatste beeld van snelwegen en is hiermee het summum van de verkeerstheorie. Dit is echter niet in lijn met dit onderzoek omdat het de begrijpelijkheid wilt handhaven. Als men echter in lijn met het doel van dit onderzoek wil blijven, is het best de focus te verschuiven naar microscopische modellen. Dit is weer een nieuw onderzoeksgebied. Een begrijpelijke uiteenzetting hierover draagt veel bij aan het begrip van bestuurders over bijvoorbeeld de werking van verkeerslichtingen of het in- en uitvoegen van voertuigen. Daarnaast kan vervolgonderzoek macroscopische modellen uitvoeriger bestuderen. Dit vereist een hogere complexiteit, maar brengt een accurater model met zich mee. Hierbij moet men denken aan het implementeren van de gevolgen van homogeniteit onder bestuurders, grote systemen, het rekening houden met voertuiggrootte of andere beperkingen van het huidige model.

Bijlage 1: plan van aanpak (PvA)

Aangezien dit onderzoek individueel wordt uitgevoerd is een taakverdeling niet van toepassing en daarom ook niet gegeven.

Fase	**Datum**	**Specificatie**
Leren en oriënteren	25/06/2022 t/m 01/08/2022	edX opleidingen in grafentheorie, verkeersstroommanagement en intelligente verkeerssystemen afronden ter voorbereiding op het onderzoek.
Leren en verdiepen	01/08/2022 t/m 17/08/2022	Het lezen van wetenschappelijke artikelen die een dieper beeld geven over de eerder opgedane kennis uit de edX opleidingen. Tevens wordt de kennis uit deze fase, maar ook vorige de fase uiteengezet in een literatuuronderzoek.
Verdiepen en toepassen	10/08/2022 t/m 17/08/2022	Inlezen in het programmeren van zowel grafentheorie als verkeersstromen en dit toepassen in pilotexperimenten. Deze worden als opstapje naar het uiteindelijke experiment gebruikt.
Toepassen	17/08/2022	De mogelijkheden van empirische data ontdekken en kijken hoe dit aansluit bij de pilotexperimenten en de ongedane kennis. Volgens wordt de onderzoeksrichting gefinetuned naar aanleiding van de opgedane kennis en de onderzoeksmogelijkheden die door het *leren, verdiepen* en *toepassen* bekend zijn geworden.
Toepassen en onderzoeken	17/08/2022 t/m 25/08/2022	Alle bevinden uit het theoretisch kader en de pilotexperimenten wordt omgezet naar pseudocode die de werking van het wiskundig model beschrijft.
Bespreken	22/08/2022 t/m 26/08/2022	Met de begeleider alles wat tot nu toe bereikt is bespreken. Hierbij ligt de nadruk op het opzetten van een uiteindelijk onderzoeksinstrument (d.w.z. het bespreken van de pseudocode).
Toepassen en Onderzoeken	25/08/2022 t/m 27/08/2022	De door bespreking aangepaste pseudocode tot een werkend model uitschrijven in Python.
Onderzoeken en optimaliseren	27/08/2022 t/m 30/08/2022	Het model testen door middel van vier testexperimenten zodat een juiste werking gegarandeerd wordt en tevens de werking van het model geoptimaliseerd wordt.
Onderzoeken, optimaliseren en concluderen	30/08/2022 t/m 03/09/2022	Het uitvoeren van vier casestudies waarbij de data afkomstig is van het Nederlandse verkeersnetwerk. Tevens wordt onderzocht of de verkregen resultaten overeenkomen met de empirische data. Congestie in de casestudies worden hierbij geanalyseerd waardoor de kenmerken van congestie in het model duidelijk worden.
Bespreken	02/09/2022	Het inleveren van het plan van aanpak.
Onderzoeken, optimaliseren en concluderen	03/09/2022	Gebieden met congestie optimaliseren door het toepassen de intelligente verkeersmanagementsystemen. Deze zijn al eerder uitgeschreven, getest en verwerkt.
Onderzoeken	04/09/2022	Het uitvoeren van een herhalingsexperiment om de validiteit van het onderzoek te controleren.
Concluderen	05/09/2022 t/m 10/09/2022	Alle bevindingen samenvoegen tot een uiteindelijke conclusie over zowel de kenmerken en plekken van congestie als de mogelijke (procentuele) verbetering die hierbij wordt bewerkstelligd.
Schrijven	05/09/2022 t/m 20/09/2022	Het begrijpelijk uitschrijven van het geprogrammeerde model, de testexperimenten, de casestudies, de optimalisering en de getrokken conclusies. Ook wordt de rest van het verslag geschreven (d.w.z. inleiding, discussie, enzovoort).
Checken, afronden en verbeteren	20/09/2022 t/m 15/11/2022	De puntjes op de i zetten voordat de voorlopige versie wordt ingeleverd. Er moet gegarandeerd worden dat er geen enkel onderdeel mist. Indien er iets mist moet dit worden toegevoegd. Daarnaast moet de juistheid en spelling gecheckt worden.
Voorlopige inlevering	Voor 15/11/2022	Het inleveren van de voorlopige versie.
Afronden en verbeteren	15/11/2022 t/m 22/12/2022	Verbeteringen van de voorlopige versie toevoegen.
Inleveren	22/12/2022	Inleveren van de uiteindelijke versie.

Bijlage 2: logboek en reflectie

logboek

Het onderstaande logboek toont de stappen die zijn ondernomen gedurende deze studie en de bevindingen die gedurende het hele proces zijn gedaan. Wanneer langdurig met een homogene tijdsverdeling aan een onderwerp is gewerkt, is dit in een langere periode aangegeven om onnodige complexiteit te verminderen.

Datum	**Tijd (in uren)**	**Plaats**	**Fase (uit plan van aanpak) en bijbehorende activiteit**	**Reflectie**
25/05/2022 t/m 28/06/2022	18,0	Thuis	Leren en oriënteren: edX opleiding over grafentheorie.	De edX opleiding bracht helderheid over de werking van grafentheorie. Het is van belang dat deze theoretische ideeën in de praktijk toe te passen en de stof beter te begrijpen. Daarom wordt hier een van de pilotexperimenten op gericht.
26/06/2022	2,0	Thuis	Oriënteren: voorbereiden van introductiedag: het schrijven van een introductie, motivatie, relevantie en de voorlopige onderzoeks- en deelvragen.	Het bracht helderheid over de mogelijke onderzoeksrichtingen binnen grafentheorie; verkeerstheorie werd namelijk gevonden. Het opstellen van onderzoeks- en deelvragen was succesvol.
28/06/2022	2,0	School	Oriënteren: Introductiedag: bespreken en aanpassen van voorlopige ideeën.	Het is een ingewikkeld onderzoeksgebied waar honderden uren literatuurstudie voor nodig zijn. De eerder opgestelde deel- en hoofdvragen bleken te voldoen aan de doelen van het onderzoek.
29/06/2022 t/m 30/06/2022	15,0	Thuis	Leren en oriënteren: start aan edX opleiding over verkeersmanagement en intelligente verkeerssystemen.	De complexiteit van de achterliggende wiskunde is te hoog. Er is veel tijd nodig om alle concepten volledig te begrijpen. Het is wel relevant voor het uiteindelijke onderzoek.
04/08/2022 t/m 07/08/2022	20,0	Thuis	Leren en oriënteren: edX opleiding over verkeersmanagement en intelligente verkeerssystemen afronden.	Het niveau nam af doordat het zich ging richten op de concepten van verkeersmanagement in plaats van achterliggende wiskunde. Verdere verdieping in de relevante onderwerpen uit de opleiding is genoodzaakt. Tevens moet de belangrijkste stof in een pilotexperiment worden toegepast om het volledig te begrijpen.
08/08/2022 t/m 09/08/2022	4,0	Thuis	Leren en verdiepen: het lezen en markeren van informatie in wetenschappelijke artikelen.	Door de eerdere studie zijn wetenschappelijke artikelen te begrijpen. Ze zijn wel complex, maar nuttig voor het onderzoek.
10/08/2022 t/m 11/08/2022	4,0	Thuis	Oriënteren en verdiepen: deelvragen aanpassen op opgedane kennis; document aanmaken waarbij alle te bespreken onderwerpen en te gebruiken bronnen bij de deelvragen staan omschreven.	Door alle eerdere studie is het genoodzaakt deelvragen aan te passen. Daarnaast is alle opgedane kennis geordend waardoor duidelijkheid in de te bespreken stof wordt geschept.
12/08/2022 t/m 16/08/2022	22,0	Thuis	Leren en verdiepen: gedurende vijf dagen elke dag ongeveer vier uur besteedt aan het uitwerken van de opgedane kennis in een theoretisch kader van de eerste vier bronnen.	Het schrijven helpt bij het beter begrijpen van de stof. De complexiteit van de stof maakt dat grafieken en tabellen niet online beschikbaar zijn. Het handmatig maken van deze kosten tijd. Ditzelfde gelden voor het invoeren van alle wiskunde functies waarvoor LaTeX geleerd is.
17/08/2022	5,0	Thuis	Verdiepen en toepassen: inlezen in de mogelijkheden op het gebied van open data en het opstellen van eisen aan het onderzoeksgebied. Dit wordt allebei uitgeschreven.	Na langdurig open datasources te vergelijken bleek NDW veruit de beste kandidaat te zijn. Daarnaast was het uitlijnen van een onderzoeksgebied nuttig. Ten eerste biedt dit duidelijk aan de lezer en bespaart het later tijd wanneer locaties voor casestudies worden gekozen.
18/08/2022 t/m 19/08/2022	12,0	Thuis	Verdiepen en toepassen: vijfde bron over CTM toevoegen aan het theoretisch kader en het programmeren en uitvoeren van een bijbehorend pilotexperiment.	Recente bronnen over CTM blijken te ingewikkeld te zijn. Het uitwerken in het theoretisch kader helpt in verhogen van het begrip van de stof. Het later uitgevoerde pilotexperiment toont de juistheid van

				de kennis aan en geeft een opstapje voor de rest van onderzoek en het uiteindelijke model.
20/08/2022 t/m 23/08/2022	14,0	Thuis	Toepassen en onderzoeken: het schrijven van de pseudocode.	Er zijn tegen veel problemen aangelopen. Het toevoegen van in- en opritten bleek complexer te zijn dan gedacht. Veel kleine schetsjes waren genoodzaakt, wat veel tijd vergde. Uiteindelijk redelijk tevreden met het resultaat. Het uitschrijven van de optimaliserende algoritme bleek gelukkig eenvoudiger dan gedacht.
24/08/2022	2,5	Thuis	Onderzoeken: downloaden en verwerken van data van casestudie 1 en het programmeren van het algoritme dat data uit csv-bestand leest.	Downloaden en verwerken van de data blijkt een tijdsintensief proces te zijn. Het programmeren van het algoritme dat de gegevens uit de csv omzet naar de juiste format voor in het model ging soepel. Dit is immers al vaker voor andere projecten gedaan.
25/08/2022	0,5	School	Bespreken: al het werk dat toe nu toe gedaan is, is in grote lijnen besproken met de begeleider. Tevens is het uiteindelijke onderzoek in detail besproken.	Wat oorspronkelijk gezien werd als pilotexperiment was voldoende voor het uiteindelijke experiment. De pseudocode is kort aan bod gekomen door zijn complexiteit. De doelen en stappen van het uit te voeren onderzoek zijn beter gedefinieerd. Het bleek dat het onderzoek al verder was dan gedacht.
25/08/2022 t/m 26/08/2022	11,5	Thuis	Toepassen en onderzoeken: de grote lijnen van het model zijn geprogrammeerd.	De pseudocode is omgezet naar een wiskundig model. Naast een aantal typefouten die pas laat in het programma gevonden werden verliep het soepel. De pseudocode was goed in orde en produceerde dus een werkend model. Enkel bleek dat flink wat fouten in de complexere onderdelen van de pseudocode zaten. Enkele onderdelen misten in de pseudocode, deze zijn toegevoegd.
27/08/2022	1,5	Thuis	Toepassen en optimaliseren: er zijn verschillende optimalisaties aangebracht.	Verschillende fouten in het model zijn verwijderd. Bepaalde onderdelen zijn in het geheel verbeterd. De code bleek echter geen veel omvattende fouten te kennen wat positief is.
28/08/2022	5,0	Thuis	Toepassen, onderzoeken en optimaliseren: casestudie 1 is getest. Casestudie 2 is gedownload en verwerkt.	Er bleken veel toch fouten in beide casestudies te zitten. Ze kwamen immers in geen enkel mate overeen met de werkelijkheid. Er moeten testexperimenten naar alleen het invoegen of uitvoegen van in- of opritten of snelwegen gedaan worden om het model juist te kalibreren.
29/08/2022	6,0	Thuis	Toepassen en optimaliseren: testexperimenten met een aantal CTM's van 10 tot 15 cellen. Hierdoor is in- en uitvoegen, maar ook RM, getest.	Veel fouten zijn aangepast en een goede werking van de geteste scenario's is gegarandeerd. Enkel is ondervonden dat één snelweg ook in verschillende andere snelwegen kan splitsen. Dit moet nog worden toegevoegd. De fouten uit de vorige casestudies zijn verholpen door de verbeteringen die de testen brengen. Daarnaast is ondervonden dat het model afhankelijker is van de data als verwacht. Kleine veranderingen hebben grote gevolgen.
30/08/2022	4,5	Thuis	Toepassen, optimaliseren en onderzoeken: casestudie 3 gedownload en getest. Code die splitsende snelwegen beschrijft, is geprogrammeerd	Voor het toevoegen van de functie was veel planning nodig. Het coderen van de nieuwe code verliep echter voorspoedig. Het downloaden en verwerken van de data verliep naar behoren.
31/08/2022	7,5	Thuis en school	Schrijven en onderzoeken: aantekeningen van logboek en plan van aanpak zijn uitgeschreven. Lay-out van verslag is opgesteld. Casestudie 4 is gedownload en verwerkt	De laatste casestudie bestaat uit een model van meer dan 50 cellen, dit maakt het complex. Al kosten het uren, er zijn geen problemen voorgevallen. Het uitschrijven van het verslag verliep voorspoedig.
01/09/2022	4,0	Thuis en school	Schrijven en onderzoeken: het uitschrijven van de methodologie en uitvoeren van casestudies 1 en 2.	Problemen. Als men te snel vooruit wil, gaat het alleen maar langzamer. Tijdens het uitvoeren de casestudies kwamen nog veel fouten naar boven.

				Veel van de schattingen blijken incorrect of de verkregen data is niet logisch. Er moet een stap terug gedaan worden. De methodologie, werking van het gehele model en testscenario's worden eerst uitgeschreven. Zo wordt het begrip van de stof verhelderd en worden fundamentele fouten gevonden. Dit wijkt af van het plan van aanpak (zie *Bijlage 1*), maar een stap terug is nodig om tot juiste resultaten te komen.
02/09/2022	4,0	Thuis en school	Schrijven, toepassen en onderzoeken: uitschrijven van het eerste pilotexperiment over grafentheorie en tekst toevoegen aan pilotexperiment over verkeerstheorie. Het inleveren van het plan van aanpak.	De pilotexperimenten zijn afgerond. Ze kennen elk verbeterpunten die benoemd zijn.
03/09/2022	3,5	Thuis	Schrijven, toepassen en onderzoeken: fout in het model is gevonden en opgelost. De fout is fundamenteel, maar het verbeterd weinig. Werking model is deels uitgeschreven.	Het uitschrijven van de werking van het model heeft geholpen bij het vinden van een fout. De hoeveelheid cellen die een cel output werd berekend door middel van de v_f in tegenstelling tot de werkelijke snelheid op een bepaald moment van de cel. Dit vereiste een aantal nieuwe functies en variabelen, maar is essentieel.
04/09/2022	5,5	Thuis	Schrijven, toepassen en onderzoeken: tijdens het bespreken vier fouten in het model gevonden en opgelost. Het gehele model inclusief optimalisering als RM en VSL is besproken. Het model is uitvoerig getest.	Men mag nu concluderen dat alle fouten uit het model verwijderd zijn. Het werkt naar behoren. De belangrijkste conclusie is dat het model gevoelig is. Een kleine afwijking kan direct tot kritieke omstandigheden leiden waarna het model geen haar meer meegeeft en het de volledige restanten van tijd geblokkeerd blijft. Data moet dus even secuur als het model worden geanalyseerd om juiste resultaten te verkrijgen. Er is deze week 35 uur aan het onderzoek gewerkt.
05/09/2022	2,5	Thuis	Schrijven, onderzoeken en toepassen: onderzoeken en schrijven van laatste twee onderdelen van Hoofdstuk 4.	De complexiteit bepalen was moeilijker dan gedacht. Voorderest verliep alles voorspoedig.
07/09/2022	5,0	Thuis	Schrijven, onderzoeken en toepassen: uitprogrammeren, uitvoeren en uitschrijven van alle testexperimenten.	Het programmeren zorgde voor enkele problemen. Naast 2 significante fouten waren de fouten vooral te vinden op het gebied van de datakeuze.
08/09/2022	4,0	Thuis	Schrijven en onderzoeken: uitvoeren en uitschrijven casestudie 1 en de helft van casestudie 2.	Aangezien de data voor alle casestudies al was voorbereid en geprogrammeerd verliep dit proces voorspoedig. Het model werkte naar behoren en fouten in de data waren eenvoudig te vinden.
09/09/2022	4,0	Thuis	Schrijven en onderzoeken: uitvoeren en uitschrijvende helft van casestudie 2 en start van casestudie 3.	De systemen worden steeds complexer en vergde daarom meer tijd. Het model lieten realistische resultaten zien.
10/09/2022	4,0	Thuis	Schrijven en onderzoeken: uitvoeren en uitschrijven van laatste deel casestudie 3 en start van casestudie 4.	Beide modellen werkte onder condities met een vrijestroomsnelheid, maar niet onder condities met congestie. Mogelijk moet er minimale snelheid worden toegevoegd. Zo kunnen er geen opstoppingen ontstaan.
11/09/2022	3,0	Thuis	Schrijven en onderzoeken: uitvoeren en uitschrijven van casestudie 3 en 4.	Veel geprobeerd, maar weinig werkte. Het is ingewikkeld om congestie in complexe systemen te modelleren door de gevoeligheid van het model. Later op de dag de problemen deels opgelost. Deze week is er 26,5 uur aan het onderzoek gewerkt.
12/09/2022	1,0	Thuis	onderzoeken: implementatie van VSL op casestudie 4.	Er blijven veel problemen. Enkele verbeteringen zijn aangebracht waardoor het model nu wel voertuigen van het begin naar het einde verplaatst.
13/09/2022	1,5	Thuis	Onderzoeken en toepassen: afronden casestudies en plannen algoritme die snelheden meet en visualiseert.	Alle casestudies zijn nu uitgevoerd. Achteraf, zijn de casestudies naar behoren afgerond. Het vinden van kleine fouten kosten veel tijd, maar er is niet tegen desastreuze fouten aangelopen, wat de werking van het model bevestigd. Het is van belang dat resultaten van het model met de werkelijke gegevens vergeleken kunnen worden.

				Het opstellen van een algoritme liep nog op niks uit omdat er te snel gestart werd met programmeren in plaats van plannen.
14/09/2022	1,0	School	Toepassen: gezocht naar een waarde om het verschil tussen empirische gegevens en gegevens uit het model met elkaar te vergelijken.	Er werd gedacht dat er eerder opgedane kennis gebruikt kon worden. Dit bleek echter niet geschikt. Later is er een werkende oplossing bedacht.
15/09/2022	2,0	Thuis	Toepassen, schrijven en onderzoeken: waarde voor het verschil tussen de empirische data en resultaten van het model geïmplementeerd. Dit is toegepast op casestudie 1 en 3.	In casestudie 1 en 3 levert het juiste resultaten op. In casestudie 2 en 4 loopt men tegen problemen aan.
16/09/2022	2,5	Thuis	Onderzoeken en schrijven: verschil tussen empirische data en het model berekenen en beschrijven voor casestudie 2 en 4.	Problemen in casestudie 4 zijn opgelost. Kosten veel tijd. In casestudie 2 zijn de problemen nog niet gevonden. Het lijkt hier op een fundamenteler probleem.
17/09/2022	1,0	Thuis	Onderzoeken: downloaden en omvormen data voor herhalingsexperiment.	Verliep voorspoedig.
18/09/2022	4,5	Thuis	Onderzoeken en schrijven: problemen in casestudie 2 opgelost en herhalingsexperiment uitgevoerd.	Het lijkt er op dat alles nu naar behoren functioneert, de problemen in casestudy 2 waren minder groot dan gedacht. Deze week is 13,5 uur aan het onderzoek gewerkt.
19/09/2022	3,0	Thuis	Schrijven: schrijven van conclusie en verbeteren opmaak (terugkerende fouten zoals het uitschrijven van getallen en punten in plaats van komma's in getallen zijn weggewerkt).	Na kortstondig het schrijven van conclusie te onderzoeken, een kort, maar krachtige conclusie geschreven. Het wegwerken van interpunctiefouten is langdradig en saai.
20/09/2022	3,5	Thuis	Schrijven: onderzoek naar schrijven van discussie. Opzet schrijven en start aan uiteindelijke discussie.	De complexiteit van de methodiek maakt het schrijven van een juiste discussie ingewikkeld. Het onderzoeken van het schrijven van een discussie en het opstellen van een schrijfplan was nodig. Vandaag stond in het plan van aanpak (zie *Bijlage 1*), dat het schrijven van de tekst afgerond moest zijn. Dit moet echter met ongeveer een week verschoven worden. Er zijn namelijk problemen ontstaan bij het uitvoeren van de casestudies en men liep vast bij het programmeren. Naast dat dit tijd kosten, leiden dit ook tot verlies aan motivatie. Hierdoor kregen andere vakken de voorkeur. Er is niet van tijdnood spraken.
21/09/2022	2,5	Thuis	Schrijven: afronden van discussie. Onderzoek naar schrijven van inleiding en maken uitgebreide opzet.	Het maken van een goede opzet blijkt tijd te besparen en verhoogt het resultaat. Daarom is dit ook voor de inleiding gedaan.
22/09/2022	2,0	Thuis	Schrijven: uitschrijven inleiding.	Verliep voorspoedig.
23/09/2022	1,5	Thuis	Schrijven: opzetten schrijfplan voor samenvatting en uitschrijven samenvatting.	Verliep voorspoedig.
24/09/2022	3,0	Thuis	Checken: start aan de uiteindelijke lay-out en het verbeteren van veel voorkomende fouten.	Lay-out en andere verwante taken kosten meer tijd dan verwacht.
25/09/2022	2,0	Thuis	Checken: De opmaak van het verslag verbeteren en het verwijderen van veel voorkomende fouten.	Bij het tegenkomen van een bepaalde taalfout komt deze vaak meerdere keren voor in het onderzoek. Het wordt direct in het hele onderzoek verbeterd. Deze week 17,5 uur aan het onderzoek gewerkt.
27/09/2022	2,0	Thuis	Checken: 85 figuren in APA-stijl plaatsen.	Na kort de specifieke regelgeving te lezen verliep het naar behoren.
28/09/2022	3,0	Thuis	Checken: verwijderen van fouten door hele document en inhoudelijk checken tot hoofdstuk 1.	Verliep voorspoedig.
29/09/2022	2,5	Thuis	Checken: 69 vergelijkingen in APA-stijl zetten. Verbeteren van enkele taalkwesties en het inhoudelijk checken van hoofdstuk 1.	Verliep voorspoedig.

01/10/2022	3,0	Thuis	Checken: checken van hoofdstuk 1. Verbeteren van enkele taalkwesties.	Verliep voorspoedig.
02/10/2022	5,5	Thuis	Checken: checken van hoofdstuk 2. Verbeteren van enkele taalkwesties.	De methodologie was verouderd en niet op niveau. Er is veel tijd besteed aan het uitbreiden en verbeteren. Daarnaast zijn vaktermen toegevoegd. Subjectieve woorden als 'kunnen', 'zouden' en verschillende bijvoeglijk naamwoorden zijn uit het hele document verwijderd. Er is gemerkt dat, al weet men dat er niet op een bepaalde manier geschreven moet worden, dit toch gebeurt omdat de gedachten meer bij de inhoud dan bij de taal liggen. Verbeteren van taalfouten kost hierdoor veel tijd. Deze week is er 18 uur aan het onderzoek gewerkt.
03/10/2022 t/m 09/10/2022	5,5	Thuis	Checken: checken van hoofdstuk 3 tot hoofdstuk 6.	Gezien dat toetsweek dicht bij komt, is er weinig tijd besteed aan het onderzoek. Het onderzoek loopt daarentegen goed op schema en het halen van de deadline is gegarandeerd. De focus op andere vakken leggen is dus logisch. Uren worden per week genoteerd omdat er maar voor korte periodes aan het onderzoek gewerkt wordt en de keuzen en bevindingen hiervan niet besproken hoeven te worden.
09/10/2022 t/m 16/10/2022	3,0	Thuis	Checken: checken van hoofdstuk 6.	Wanneer er tijd over is, wordt het verslag gecontroleerd. Inhoudelijk blijken er weinig fouten in het verslag te zitten.
17/10/2022 t/m 21/10/2022	7,5	Thuis	Checken: checken van hoofdstuk 7, 8 en 9. De laatste hand aan het verslag leggen. Eerste versie ingeleverd.	Verliep voorspoedig.
12/11/2022	4,0	Thuis	Checken: checken verslag. Ordenen van bestanden en uploaden op GitHub.	Verliep voorspoedig.
5/12/2022	1,0	Thuis	Checken: de feedback op de eerste versie is ontvangen en verwerkt.	De feedback bestond uit een aantal kleine punten die nu verwerkt zijn in het onderzoek. Dit verliep voorspoedig.
12/12/2022	1,0	Thuis	Checken: checken t/m hoofdstuk 1.	Verliep voorspoedig. Redelijk veel fouten in verwijzingen naar grafieken en vergelijkingen. Dit verbeteren is tijdsintensief aangezien deze waarde allemaal aan elkaar gelinkt zijn.
14/12/2022	5,0	Thuis	Checken: checken van hoofdstuk 2 t/m hoofdstuk 4.	Verliep voorspoedig.
15/12/2022	1,5	Thuis	Checken: checken van hoofdstuk 5 samen met verbeteren andere fouten.	Verliep voorspoedig.
16/12/2022	3,0	Thuis	Checken: checken van hoofdstuk 6 t/m hoofdstuk 7.	Verliep voorspoedig.
17/12/2022	2,0	Thuis	Checken: checken van hoofdstuk 8 t/m 9 en het checken van bijlage 1 en een deel van 2.	Verliep voorspoedig.
18/12/2022	2,0	Thuis	Checken: checken van bijlage 2, 3 en 4 en het verbeteren van de lay-out.	Lay-out vereisten veel werk omdat pagina's niet meer goed op elkaar aansloten.
20/12/2022	3,0	Thuis	Checken en schrijven: schijven van reflectie, verbeteren van de lay-out, uitvoeren spellingscheck. Inleveren.	Verliep voorspoedig.

Reflectie

De verkeerstheorie was een nog onbekende studie waardoor er verwacht werd dat de literatuurstudie veel tijd zou vergen. De complexiteit van deze studie was echter nog steeds onderschat. Hierdoor kosten elk aspect van het onderzoek nog steeds meer tijd dan verwacht. Als gevolg is er het oorspronkelijke plan van aanpak niet gehaald. Dit terwijl er meer tijd aan het onderzoek is besteed dan dat oorspronkelijk gepland was. Vooral tijdens het testen van het model zijn er veel nieuwe problemen naar boven gekomen die elk onverwacht veel tijd vraagde. Aan de andere kant zijn door het oplossen van de problemen veel grenzen verlegd en is veel nieuwe kennis opgedaan. Niet alleen is door dit onderzoek kennis over de verkeerstheorie verkregen, ook de programmeervaardigheden zijn verbeterd. In vergelijking tot het programmeren van het model, was het schrijven van het verslag eenvoudig. Toch is hier veel kennis opgedaan over de elementen van een wetenschappelijk verslag. Iets wat op de universiteit van pas zal komen. Kortom, het uitvoeren van dit onderzoek was extreem leerzaam. Het onderzoeksproces werd gekenmerkt door veel hoogte- en dieptepunten, maar door vanaf moment één vol doorzettingsvermogen aan het project te werken en niet op te geven, al leek of problemen soms onoverkomelijk waren, is het onderzoek tot een succesvol einde gebracht.

Aanpak files A1-A30 bij Barneveld stap dichterbij. (2021, 24 februari). XON. https://www.xon.nu/5740-aanpak-files-a1a30-bij-barneveld-stap-dichterbij

Carlson, R. C., Papamichail, I., Papageorgiou, M. & Messmer, A. (2010). Optimal mainstream traffic flow control of large-scale motorway networks. *Transportation Research Part C: Emerging Technologies*, *18*(2), 193–212. https://doi.org/10.1016/j.trc.2009.05.014

CROW. (2022, februari). *Real Time Traffic Information Een duiding van de nieuwe RTTI gedelegeerde verordening voor wegbeheerders* (Nr. D397). https://www.crow.nl/downloads/pdf/verkeer-en-vervoer/verkeersmanagement/d397_real-time-traffic-information_nl.aspx

Daganzo, C. F. (1992). The Cell Transmission Model. Part I: A Simple Dynamic Representation Of Highway Traffic. *Transportation Research Part B: Methodological*. https://escholarship.org/uc/item/0b6612tk

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, *28*(4), 269–287. https://doi.org/10.1016/0191-2615(94)90002-7

Daganzo, C. F. (1995). The cell transmission model, part II: Network traffic. *Transportation Research Part B: Methodological*, *29*(2), 79–93. https://doi.org/10.1016/0191-2615(94)00022-r

École Polytechnique Fédérale de Lausanne, Geroliminis, N., Kouvelas, A. & Lamotte, R. (2022, 29 juli). *Intro to Traffic Flow Modeling and Intelligent Transport Systems*. edX. Geraadpleegd op 12 juli 2022, van https://www.edx.org/course/intro-to-traffic-flow-modeling-and-intelligent-tra#:%7E:text=Intelligent%20Transport%20Systems-,Learn%20how%20to%20describe%2C%20model%20and%20control%20urban%20traffic%20congestion,mobility%20in%20cities%20and%20highways.

Frejo, J. R. D., Papamichail, I., Papageorgiou, M. & Schutter, B. D. (2018). A new macroscopic model for Variable Speed Limits. *IFAC-PapersOnLine*, *51*(9), 343–348. https://doi.org/10.1016/j.ifacol.2018.07.056

Hegyi, A., De Schutter, B. & Hellendoorn, H. (2005). Model predictive control for optimal coordination of ramp metering and variable speed limits. *Transportation Research Part C: Emerging Technologies*, *13*(3), 185–209. https://doi.org/10.1016/j.trc.2004.08.001

Hollandse Hoogte / Spaarnestad Photo. (z.d.). *Autoloze zondag 1973 in verband met de oliecrisis.* [Foto]. Zo groot is de kans dat de autoloze zondag er komt. https://www.ad.nl/binnenland/zo-groot-is-de-kans-dat-de-autoloze-zondag-er-komt~a99edd18/

Institut Mines-Télécom, Gripon, V., Meyer, P. & Farrugia, N. (2022, 1 mei). *Advanced Algorithmics and Graph Theory with Python*. edX. Geraadpleegd op 1 juli 2022, van https://www.edx.org/course/advanced-algorithmics-and-graph-theory-with-python?index=product&queryID=35691bae886f505d21b6bc99a98126ae&position=1

Khondaker, B. & Kattan, L. (2015). Variable speed limit: an overview. *Transportation Letters*, *7*(5), 264–278. https://doi.org/10.1179/1942787514y.0000000053

Knoop, V.L., Duret, A., Buisson, C. & B. Van Arem. (2010). Lane distribution of traffic near merging zones influence of variable speed limits. In Proc. of the 13th International IEEE Conference on Intelligent Transportation Systems (ITSC) (pp. 485). Madeira, Portugal.

Learn Advanced Algorithms and Data Structures with Python. (z.d.). Codecademy. Geraadpleegd op 1 mei 2022, van https://www.codecademy.com/learn/learn-advanced-algorithms-and-data-structures

Learn Data Structures and Algorithms with Python. (z.d.). Codecademy. Geraadpleegd op 1 april 2022, van https://www.codecademy.com/learn/learn-data-structures-and-algorithms-with-python

Lege wegen: unieke foto's! (2022). Veilig Verkeer Nederland. https://vvn.nl/nieuws/lege-wegen-unieke-fotos

Luttinen, T. (1996, januari). *1: Time-space diagram of vehicle trajectories* [Foto]. https://www.researchgate.net/figure/Time-space-diagram-of-vehicle-trajectories_fig2_285485763

Master Statistics with Python. (z.d.). Codecademy. https://www.codecademy.com/learn/paths/master-statistics-with-python

Ministerie van Infrastructuur en Waterstaat. (2020, 20 januari). *6 vragen over de file*. Nieuwsbericht | Kennisinstituut voor Mobiliteitsbeleid. Geraadpleegd op 22 september 2022, van https://www.kimnet.nl/actueel/nieuws/2020/01/21/zes-vragen-over-de-file#:%7E:text=Hoe%20lang%20staan%20we%20in,gemiddeld%2043%20minuten%20vertraging%20op.

Ministerie van Infrastructuur en Waterstaat. (2021, 7 april). *Afname luchtvervuiling tijdens coronacrisis*. KNMI Jaaroverzicht. Geraadpleegd op 8 september 2022, van https://magazines.rijksoverheid.nl/knmi/knmi-jaaroverzicht/2020/01/minder-luchtvervuiling-door-corona

Ministerie van Infrastructuur en Waterstaat. (2022a). *Nationaal Dataportaal Wegverkeer*. NDW. Geraadpleegd op 17 augustus 2022, van https://www.ndw.nu/

Ministerie van Infrastructuur en Waterstaat. (2022b, februari 3). *A28/A1: project knooppunt Hoevelaken*. Rijkswaterstaat. Geraadpleegd op 9 september 2022, van https://www.rijkswaterstaat.nl/wegen/projectenoverzicht/planuitwerking-knooppunt-hoevelaken#hoe

Ministerie van Infrastructuur en Waterstaat. (2022c, mei 12). *Nationaal Dataportaal Wegverkeer - Nationaal Dataportaal*

Wegverkeer. Geraadpleegd op 22 september 2022, van https://www.ndw.nu/

Miessler, D. (2019b, september 14). Big-O Notation Explained. Daniel Miessler. Geraadpleegd op 5 december 2022, van https://danielmiessler.com/study/big-o-notation/

MIRT, Gemeente Barneveld. (2018, 13 juli). *MIRT-ONDERZOEK AANSLUITING A1/A30 BARNEVELD*. Gelderland. Geraadpleegd op 8 september 2022, van https://media.gelderland.nl/MIRT_Onderzoek_aansluiting_A1_A30_Barneveld_bee032c933.pdf

MITx. (2022). *Introduction to Differential Equations*. edX. https://www.edx.org/course/introduction-to-differential-equations-2

NDW. (2021–2022). *NDW Dexter* [Dataset].

Papageorgiou, M., Kosmatopoulos, E. & Papamichail, I. (2008). Effects of Variable Speed Limits on Motorway Traffic Flow. *Transportation Research Record: Journal of the Transportation Research Board*, *2047*(1), 37–48. https://doi.org/10.3141/2047-05

Papageorgiou, M. & Kotsialos, A. (2003). Freeway ramp metering: an overview. *IEEE Transactions on Intelligent Transportation Systems*, *3*(4), 271–281. https://doi.org/10.1109/tits.2002.806803

Papamichail, I., Kampitaki, K., Papageorgiou, M. & Messmer, A. (2008). Integrated Ramp Metering and Variable Speed Limit Control of Motorway Traffic Flow. *IFAC Proceedings Volumes*, *41*(2), 14084–14089. https://doi.org/10.3182/20080706-5-kr-1001.02384

Rijkswaterstaat. (2012–2021). *INtensiteit WEgVAkken (INWEVA)* [Dataset]. https://maps.rijkswaterstaat.nl/dataregister/srv/dut/catalog.search#/metadata/f58eacc9-ca69-487a-a53b-11efad0bbba0

Rijkswaterstaat. (2022, 22 maart). *Filezwaarte neemt toe, nog niet op de helft van het niveau van voor corona*. Geraadpleegd op 22 september 2022, van https://www.rijkswaterstaat.nl/nieuws/archief/2022/03/filezwaarte-neemt-toe-nog-niet-op-de-helft-van-het-niveau-van-voor-corona#:%7E:text=Het%20aandeel%20files%20door%20werkzaamheden,verklaren%20door%20meer%20ongeplande%20werkzaamheden.

Satellietfoto knooppunt A1/A30. (z.d.). Google. Geraadpleegd op 8 september 2022, van https://www.google.com/maps/

Satellietfoto knooppunt Hoevelaken. (z.d.). Google maps. Geraadpleegd op 9 september 2022, van https://www.google.com/maps/

Satellietfoto knoopunt Maanderbroek. (z.d.). Google maps. Geraadpleegd op 8 september 2022, van https://www.google.com/maps/

Satellietfoto raNDWeg Eindhoven. (z.d.). Google maps. Geraadpleegd op 9 september 2022, van https://www.google.com/maps/

Tijdlijn van coronamaatregelen. (2022). RIVM. Geraadpleegd op 8 september 2022, van https://www.rivm.nl/gedragsonderzoek/tijdlijn-maatregelen-covid

Wikipedia. (z.d.-a). *A drawing of a graph.* [Foto]. https://en.wikipedia.org/wiki/Graph_theory#/media/File:6n-graf.svg

Wikipedia. (z.d.-b). *Demonstration of Dijkstra's algorithm on a small graph, showing two relaxation operations.* [Foto]. https://nl.wikipedia.org/wiki/Kortstepad-algoritme#/media/Bestand:Dijkstra's_algorithm.svg

Zegeye, S.K., De Schutter, B., Hellendoorn, H. and Breunesse, E. (2010). "Reduction of Travel Times and Traffic Emissions Using Model Predictive Control." Proceedings of the 2009 American Control Conference, St. Louis, Missouri, pp. 5392-5397.

Bijlage 4: wiskundig model en data

Op de onderstaande website zijn de rauwe code en de data van dit gehele onderzoek te vinden:

https://github.com/StandeHaas/File_File_en_nog_eens_File