Touring Machines
Traffic Simulation

Sinem Ertem, Martijn Knegt & Gijsbert Nutma

Hogeschool Utrecht
2019_TAAI-V2ST-10_2_V
Periode B (2019/20)

Inhoudsopgave

1. Inleiding

Voor opdracht 5 van Simulation Tooling zullen we een ABM (Agent-Based Model) gaan bouwen voor het verkeer zoals we dit ook in de werkelijkheid kennen. Wij zullen ons vooral gaan focussen op verkeerscongestie. Dit is een tijdelijke verstopping in het verkeersnetwerk wat in ons geval veroorzaakt wordt door het plotseling remmen van een auto. Hierdoor ontstaat er een golf van vertraging en zullen de auto's op een gegeven moment tot stilstand komen. Het model wat wij gaan maken bestaat is een variatie op: '*The Nagel-Schreckenberg model*'.[1] Dit model is rond 1990 ontworpen door twee Duitse natuurkundigen K. Nagel en M. Schreckenberg.

1.2 Onderzoeksvraag

Met de kennis die we hebben opgedaan met het *Nagel-Schreckenberg model* en voorbeeldsimulaties van *NetLogo* hebben we een onderzoeksvraag geformuleerd die ons helpt bij het onderzoeken van filevorming:

"Hoe hebben drukte en maximumsnelheid effect op filevorming in het verkeer?"

Onder 'drukte' verstaan we het aantal auto's op één rijbaan. Een 'file' wordt gedefinieerd als auto's die niet meer hun originele snelheid hebben en deze niet meer kunnen bereiken.

1.3 Plan van aanpak[2]

1.3.1 Agent

De Agents in ons model bestaan alleen uit auto's. Deze hebben weer elk hun attribuut 'speed' (snelheid). De Agents beginnen met een start-speed, deze is hetzelfde als de ingestelde maximumsnelheid. De regels die de agenten verder meekrijgen zijn:

- Remmen/stoppen als de Agents te dicht bij elkaar in de buurt komen. Hiervoor hebben we een 'target-brake' die dit beïnvloed. Zodra de gebruiker van de simulatie klikt op deze button zal de target-auto (rood van kleur) op de rem gaan.
- Op één rijbaan blijven.
- Auto's kunnen niet door elkaar heen. Het is de bedoeling dat er op 1 patch van de rijbaan maar 1 auto op kan rijden omdat de auto's elkaar anders gaan overlappen.

1.3.2 Environment

Zoals hierboven al toegelicht zullen de auto's alleen op één rijbaan gaan rijden. De rijbaan bestaat uit een grid met cellen/patches. De environment heeft een oneindige rijbaan (net zoals in het *Nagel-Schreckenberg model*). In ons model zullen de agents alleen op elkaar reageren.

1.3.3 Toolkeuze

Aan de hand van het SFA model zullen we de verschillende tools: Mesa, Unity en NetLogo tegen elkaar afwegen. We zullen alleen op *Suitability* en *Feasibility* beoordelen en punten uitdelen van 1 t/m 5. De tool met de meeste punten zal onze voorkeur hebben.

[1] Bron: https://en.wikipedia.org/wiki/Nagel–Schreckenberg_model

[2] Zie *model.png* voor volledige visualisatie van de planontwikkeling

1.3.3.1 Mesa

Suitability

Tool support: De gewenste user story zou zeer waarschijnlijk gemaakt kunnen worden in Mesa voornamelijk omdat je met goede kwaliteit van alles kan visualiseren. en natuurlijk dat je het programmeert in pyhton waarin je praktisch alles kan maken als je het maar kunt bedenken. Er zit echter geen module ingebouwd die ervoor zorgt dat de Agents elkaar kunnen zien en met elkaar kunnen communiceren, bijvoorbeeld welke snelheid de andere Agent rijdt. Dat zou dan zelf gecodeerd moeten worden wat tijd zou kunnen kosten. ***Punten**: 3*

Performance efficiency: Een nadeel van Mesa is als je heel veel code hebt en/of als je code niet heel efficiënt opgebouwd hebt, de animatie in Mesa minder goed zal verlopen. Python is te lang bezig met de calculaties waardoor je animatie langzaam en stotterend kan verlopen. ***Punten**: 2*

Compatibility: Voor dit project zal er waarschijnlijk geen externe data nodig zijn maar mochten we het alsnog nodig hebben is het geen probleem omdat Mesa in python is geschreven. Hierdoor kan Mesa met praktisch alle externe data werken. ***Punten**: 4*

Feasibility

User-friendly: Bij Mesa moet je de hele simulatie maken en is er geen basis om vanaf te werken. Dit kan voor beginners best lastig zijn. ***Punten**: 1*

Technically feasible: Wij denken dat het voor ons niet mogelijk is om een goede simulatie te maken in Mesa binnen 2 weken omdat Mesa behoorlijk groot is en omdat er best wel veel coding nodig is om de simulatie werkend te krijgen. ***Punten**: 1*

1.3.3.2 NetLogo

Suitability

Tool support: De user story kan gemaakt worden in NetLogo, de tool is geschikt voor simulaties. We hebben een simulatie in NetLogo gezien die lijkt op de simulatie die wij willen maken. ***Punten**: 5*

Performance efficiency: NetLogo heeft een ‘simulation-speed’ slider waarmee de snelheid van de simulatie aangepast kan worden. Hiermee is het mogelijk om de simulatie heel snel te laten runnen. ***Punten**: 4*

Compatibility: Voor onze simulatie maken we geen gebruik van externe data maar gebruiken we slechts alle tools in NetLogo om zelf data te creëren. NetLogo heeft wel een GIS extensie die bepaalde data kan inladen, mocht je het nodig hebben. ***Punten**: 3*

Feasibility

User-friendly: Het gebruiksvriendelijke aan NetLogo is dat het al een ‘Models Library’ bevat waardoor gebruikers al gemaakte simulaties kunnen inladen en uitvoeren.Verder zijn er drie verschillende kolommen; interface waarin de simulatie zichtbaar is, waarin verdere informatie staat met uitleg en code waarin de gebruiker

kan zien hoe de simulatie is opgezet en zelf nog aanpassingen kan doorvoeren. ***Punten**: 4*

Technically feasible: Door alle elementen die onder het kopje *User-friendly* zijn genoemd is het mogelijk om binnen de deadline een eigen simulatie te maken. Dit omdat het gemakkelijk is om de tool binnen de deadline te begrijpen. ***Punten**: 4*

1.3.3.3 Unity

Suitability

Tool support: Unity is per definitie geschikt voor de user story, een baan met een object (auto) erop is te modelleren in het programma. Hier moet wel rekening gehouden worden met de soms lastige omgeving van Unity, aangezien deze meer geschikt is voor games dan simulaties. ***Punten: 4***

Performance efficiency: Unity is in principe snel genoeg om simulaties uit te voeren. Misschien kan het performance-wise lastig zijn voor Unity3D om met veel Agents te werken als daar niet specifiek op geprogrammeerd wordt. ***Punten:** 3*

Compatibility: Unity heeft C# als achterliggende programmeertaal. Met deze programmeertaal zijn veel libraries en mogelijkheden om externe data te gebruiken. ***Punten:** 4*

Feasibility

User-friendly: Bij het opstarten van Unity is de interface best overweldigend, er zijn veel mogelijkheden en dingen die aangepast kunnen worden. Dit kan het moeilijk maken om snel een werkend model te maken. ***Punten:** 2*

Technically feasible: Doordat Unity erg uitgebreid is, kan het lastig worden om deze tool binnen twee weken te leren en te gebruiken om een goede simulatie mee te maken. ***Punten:** 2*

1.3.4 Score toolkeuze

	Mesa	Unity	NetLogo
Suitability	9	11	12
Feasibility	2	4	8
Totaal punten	11/25	15/25	20/25

Door voor elk onderdeel punten toe te kennen en deze bij elkaar op te tellen zijn we tot de conclusie gekomen dat NetLogo met 20 van de 25 punten onze voorkeur heeft.

2. Design experiment

Aantal auto's:

- Het aantal auto's varieert van 5 auto's tot 50 auto's. Hier worden stappen van 1 auto genomen. We krijgen zo een aantal auto's: 5, 6, 7, 8 … 50.

Maximum snelheid:

- De snelheid varieert van 0.1 tot 1.0 met stappen van 0.1. Op deze manier krijgen we maximumsnelheden van 0.1, 0.2, 0.3 … 1.0.

Target Brake:

- De Target Brake is het 'event' in de simulatie. Dit zorgt ervoor dat de *target* auto remt naar een gewenste snelheid. Op deze manier zetten we het doel van de simulatie in gang. De Target Brake zorgt voor het zetten van de speed van de target naar 0.1. Bij het runnen van het experiment wordt dit event automatisch getriggerd op ticks = 20.

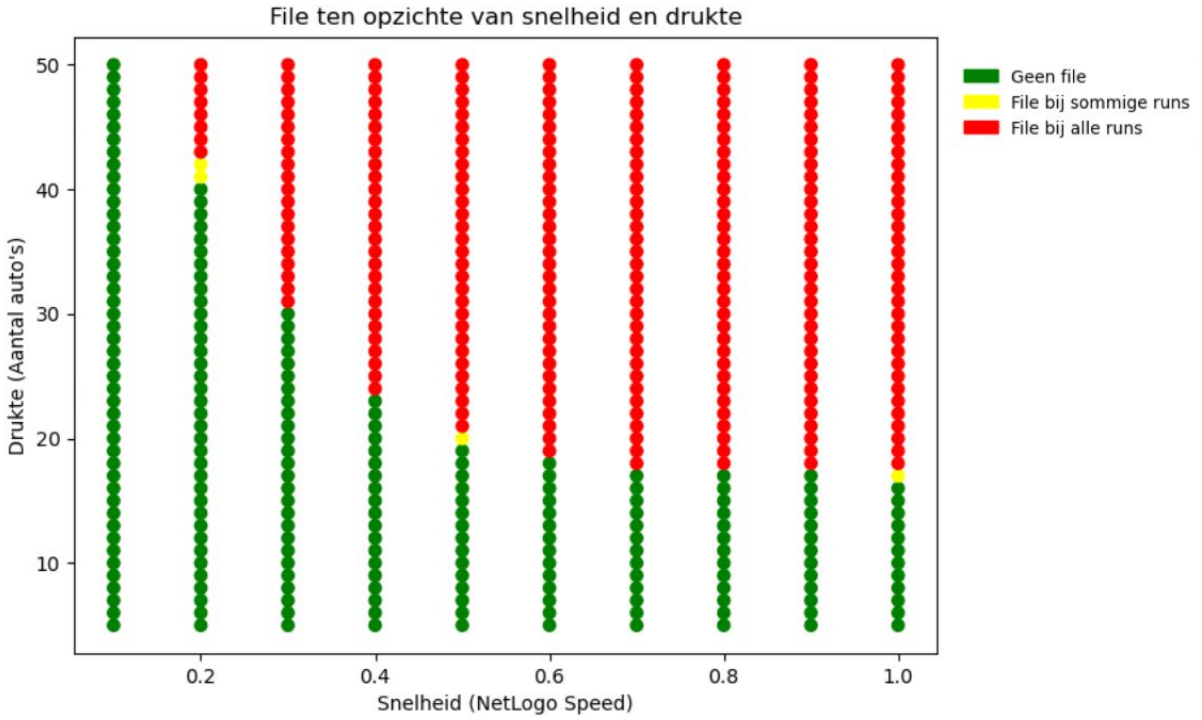
In het programma hebben wij ook de lengte van de simulatie variabel gemaakt. We hebben uiteindelijk gekozen om bij al onze settings een vaste lengte van 2000 ticks te nemen. Bij een kleine hoeveelheid auto's is het remeffect niet bijzonder groot, waardoor de auto's hun maximumsnelheid snel weer bereiken. De rest van de simulatie blijft dan gelijk (het rem-event doet zich maar één keer voor) waardoor we heel veel 'constante' data overhouden.

Bij het runnen van verschillende settings wordt elke keer data opgeslagen. Per setting wordt er 10 keer gerund, bij elke run wordt een .csv geëxporteerd met daarin de snelheid per tick van de target auto, en de minimumsnelheid van alle auto's.

Daarnaast wordt bij elke run de gemiddelde snelheid van de target opgeslagen in een .txt bestand, waardoor we per setting een .txt bestand krijgen met 10 regels, voor elke run per setting. De gemiddelde snelheid die wordt opgeslagen is niet over het hele traject, maar het gemiddelde over de ticks 1700-2000. Met deze waarde kan bepaald worden of de huidige setting resulteert in een file of niet.

3. Resultaten experiment

figuur 1: File ten opzichte van snelheid en drukte



In de bovenstaande grafiek is de verkeerssituatie weergegeven ten opzichte van het aantal auto's en de maximumsnelheid die de auto's mogen rijden.

De *groene* punten geven weer dat er voor elke run geen file ontstaat in deze setting. De vertraging die ontstaat kan zichzelf weer oplossen, aan het einde van de simulatie rijdt de target weer op de maximumsnelheid.

De *gele* punten geven aan dat in sommige van de 10 runs van deze setting een file ontstaat, de vertraging die ontstaat kan zichzelf in deze runs niet oplossen, maar dit is niet voor alle runs het geval.

De *rode* punten geven aan dat er voor elke run van de setting een file ontstaat. In geen van de 10 gevallen dat de setting gerund wordt kan de vertraging zichzelf oplossen en ontstaat er een file.

Wat opmerkelijk is bij deze grafiek is het bestaan van de gele punten. We hebben ons model zoveel mogelijk *deterministisch* gemaakt. Alle elementen die variabel zijn, worden per setting vastgezet op een waarde met behulp van sliders, andere waarden zijn voor alle settings hetzelfde (opgenomen in de code). Wij denken dat er toch nog een verschil in simulaties kan ontstaan, voornamelijk van belang op de kantelpunten (geel), door het verschil in acceleratie en deceleratie. Deze waarden zijn niet hetzelfde, en in combinatie met het feit dat elke auto voor zichzelf kijkt of een andere auto zich voor hem bevindt kan het gebeuren dat de runs per setting wat van elkaar afwijken.

4. Conclusie

Onze onderzoeksvraag zoals in het begin van het verslag staat vermeld luidt:
"Hoe hebben drukte en maximumsnelheid effect op filevorming in het verkeer?"

In de resultaten van ons experiment kunnen we een verhouding terugzien tussen het aantal auto's en de maximumsnelheid die de auto's kunnen hebben. Hierbij kunnen we zien dat hoe meer auto's de weg bevat, hoe groter het effect is op de vertraging. Hetzelfde geldt voor de snelheid van de auto's: hoe groter de snelheid, hoe groter het effect op de vertraging.

Zoals we in *figuur 1* kunnen zien, ontstaat er voor bijna elke snelheid file wat zichzelf niet meer kan oplossen. Dit hangt dan wel weer af van de hoeveelheid auto's op de weg. Bij 50 auto's is er bijvoorbeeld (bijna) altijd file omdat we zogezegd niet genoeg 'infrastructuur' hebben en de rijbaan in de simulatie te vol is met auto's waardoor er meteen file ontstaat zodra de auto ook maar een beetje zijn snelheid vermindert. Dit is precies wat in de werkelijkheid ook gebeurt als er meer auto's zijn op de weg dan dat er infrastructuur is. Dit gebeurt dan bijvoorbeeld tijdens de spits.

Tot een aantal van 17 auto's ontstaat er nooit file als we kijken naar elke combinatie met maximumsnelheid. Dat de target-auto hard remt heeft dus verder geen invloed op de rest van het verkeer. De snelheid zal even verminderd worden maar dit wordt weer snel opgelost. We kunnen dan concluderen dat dit een perfect aantal is voor het verkeer en vanaf het moment dat er meer auto's komen is er meer kans op (langdurige) filevorming.

5. Discussie

Voordat we daadwerkelijk onderzoek hadden gedaan, hadden we wel verwacht dat er uiteindelijk een file ontstaat wanneer de auto's niet meer hun eigen snelheid behouden en door een lage snelheid de rest van de weggebruikers belemmert. Het was nog de vraag bij ***hoeveel*** auto's dit ontstaat en welke factoren daadwerkelijk effect hebben.

5.1 Aanpassingen

Aanvullend aan het onderzoek wat wij hebben uitgevoerd, zou er wat aan het model aangepast kunnen worden zodat het beter naar de werkelijkheid vertaald kan worden en je betere resultaten eruit kan halen voor het onderzoek.

5.1.1 Rijbaan

Zo hebben wij zoals het *Nagel-Schreckenberg model* ook heeft, een oneindige rijbaan waarin de auto's aan de rechterkant eruit rijden en via de linkerkant weer binnenrijden. Een aanpassing zou kunnen zijn om het niet oneindig te laten lopen en te onderzoeken wat het effect hiervan is. Een loop zoals dit is namelijk niet hoe het in de werkelijkheid ook gaat.

5.1.2 Brake

Een verdere aanpassing zou de brake van de auto kunnen zijn. In ons model is het namelijk zo dat de auto meteen afremt tot een lage snelheid zonder dat deze op een rustige manier wordt verminderd zoals dat in de werkelijkheid ook gebeurt. Met een echte verkeersgedrag zouden we ook rekening moeten houden met de combinaties van snelheid, reactietijd en remweg.

5.1.3 Rijgedrag

Bij het simuleren gaan wij ervan uit dat elke auto netjes afstand houdt en de juiste maximumsnelheid blijft rijden waar mogelijk. In de realiteit zijn deze dingen niet aan de orde.

5.2 Realiteit

De simulatie is slechts een abstracte vertaling van de realiteit omdat er nog veel punten verbeterd kunnen worden zoals hierboven onder het kopje aanpassingen staat vermeld. Dit kan deels een onjuist beeld weergeven van filevorming maar voor een groot deel, als we kijken naar de parameters *aantal auto's* en *snelheid* geeft het toch interessante data omdat precies hetzelfde in de werkelijkheid ook gebeurt. Er is een statistische samenhang tussen deze twee grootheden en samen bepalen ze de langdurigheid van een verkeerscongestie.