Verkeerssimulatie onderzoeksplan

Quincey Mok, Sjoerd Beetsma, Maarten de Jeu

Inleiding

Het ontstaan van files en het optimaliseren van verkeersdoorstroom zijn problemen waar wiskundigen al ontzettend lang mee bezig zijn. De inzet van multi-agent simulaties hebben daar in recente jaren inzichten in gegeven die wiskundige formules vaak niet konden geven, door gedrag te definiëren op het niveau van individuele deelnemers aan een systeem, in plaats van op systeemniveau. Een simpel model dat hier al inzicht in kan geven is Nagel-Schreckenberg model. Deze zullen wij gebruiken om de invloed van verkeersdichtheid op het vormen van files te onderzoeken in verschillende situaties.

Onderzoeksvraag

Welke invloed heeft de verkeersdichtheid op het vormen van files?

Hypothese

De verwachting die wij stellen is dat er een verband is tussen de verkeersdichtheid en de doorstroom van het verkeer, ook wel verkeers-flow genoemd. Als de verkeersdichtheid stijgt verwachten we dat doorstroom van het verkeer zal gaan dalen. Er staan dan te veel auto's op de weg, waardoor er een file kan ontstaan. Op het moment dat er een file ontstaat verwachten we dat we een kantelpunt kunnen zien die plotselinge files kan detecteren. Dat gebeurd pas als de verkeerdichtheid een bepaald punt heeft bereikt, waardoor de doorstroom van het verkeer ineens zal dalen naar 0. Hierdoor hopen wij dat we met behulp van de simulatie dit probleem te detecteren.

Plan van aanpak

Voor het plan van aanpak maken wij gebruik van de MoSCoW methode. Het MoSCoW methode is een simpele, maar een krachtige tool die je helpt met het priotiseren van taken. De methode wordt ingedeeld in de volgende categorieën:

M - must haves:
deze eisen (requirements) moeten in het eindresultaat terugkomen, zonder deze eisen is het product niet bruikbaar;

S - should haves:
deze eisen zijn zeer gewenst, maar zonder is het product wel bruikbaar;

C - could haves:
deze eisen zullen alleen aan bod komen als er tijd genoeg is;

W - won't haves:

deze eisen zullen in dit project niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

In de MoSCoW methode hebben de kleine letters o's geen betekenis. Het is alleen bedoeld om de naam van de methode makkelijker uit te spreken. Door de taken in verschillende categorieën te plaatsen, krijg je een mooi overzicht welke taken belangrijk zijn in je product. Het is namelijk handig om te zien wat een de belangrijkste taken zijn binnen je project mocht je een Minimum Viable Product willen opleveren. Op deze manier weet men welke taken je als eerst moet oppakken of welke taken je zelfs moet schrappen wanneer er een tijd tekort is. Hierdoor leer je efficiënt met je tijd om gaan. De MoSCoW-methode wordt regelmatig toegepast bij productontwikkeling.

Voor het realiseren van onze prototype en het beantwoorden onze onderzoeksvraag hebben wij hieronder onze Must-have, Should have, Could have. In overleg met de docent hoeven we voor dit project de Won't-have in de MoSCoW-methode niet de definieren.

Must-have:

1. De mogelijkheid om de simulatie te herstarten.
2. De doorstroom van het verkeer moet worden berekend.
3. De verkeersdichtheid (density) moet worden berekend.
4. Een omgeving bestaande uit een rechte weg met een periodieke randvoorwaarden (zodat deze oneindig loept).
5. Gediscretiseerde tijd in tijdstappen, waar bij elke tijdstap de agents(voertuigen) de omgeving(het verkeer) waarnemen en daarop gebaseerd (samen met hun eigen interne staat) een handeling ondernemen.
6. De mogelijkheid de staat (snelheid) per individuele agent verkrijgbaar maken, terwijl de simulatie draait.
7. Een GUI om de simulatie te visualiseren.
8. De mogelijkheid resultaten uit de simulatie op te slaan.
9. De mogelijkheid om het gedrag van de agent te simuleren en een omgeving voor de agent bouwen.
10. De agent moet remmen als een agent voor hem te dicht op hem zit.
11. De agent moet de maximale snelheid aanhouden als er geen agent voor hem zit.
12. De agent moet een kans hebben om willekeurig af te remmen.

Should haves:

1. De mogelijkheid om de simulatie te herstarten met verschillende instellingen.
2. Parameters instelbaar maken vanuit de GUI.
3. Resultaten live tonen in de GUI.
4. Resultaten vergelijken met data uit de echte wereld.

Could haves:

1. De mogelijkheid het verbeterde model als voorgesteld door Chen, H.-J (2001) naast het standaard model te gebruiken, en de resultaten te vergelijken.
2. De simulatie kan meerdere rijbanen simuleren.
3. De agent kan ritsen op een andere rijbaan.
4. De simulatie kan kruispunten simuleren.
5. De simulatie kan verkeerslichten simuleren.
6. Het gebruik van Data-Science om meer inzichten in de resultaten te verkrijgen.

Tool keuze SF-model

Netlogo, Mesa en Unity zijn drie verschillende tools die gebruikt kunnen worden voor het opstellen van *ABS* (Agent based simulations). Om een keuze uit deze tools te maken voor het gebruik bij ons onderzoek hebben we de voor-en nadelen in kaart gebracht en is er een SF analyse opgesteld. Een SF(A) matrix is een hulpmiddel om verschillende strategieën/keuzemogelijkheden te analyseren. SF staat voor het volgende:

Suitability (geschiktheid):
De mate waarin de optie geschikt is voor de doelstelling.
Feasibility (haalbaarheid):
De mate waarin de optie haalbaar is.

Hierbij wordt er gekeken naar suitability, geschiktheid (S) en feasibility (F).
Elk van deze 3 tools krijgt een score per onderdeel om zo te bepalen welke tool het best geschikt is voor het onderzoek.

Netlogo:
Netlogo is een gebruiksvriendelijke all-in-one 2D simulatie tool. Een ABS in netlogo is te bevatten in de vorm van turtles, patches, links en de Observer. Deze 4 onderdelen bieden een high-level manier om simpel een simulatie op te zetten.

Voordelen:
- High level taal (Eenvoudig).
- Simulatie in simpele concepten zoals (turtles, patches, links en de observator).
- Ingebouwde live-grafieken over de simulatie.
- GUI ingebouwd.

Nadelen:
- Bugs/errors lastig te ontcijferen
- High level taal (beperkend).
- Performance.
- Mist standaard programmeertools.
- Werkt alleen synchroon.
- Ingebouwde IDE werkt niet fijn (geen syntax highlighting).
- Agents kunnen niet communiceren.

Mesa:
Mesa is een simulatie-library in Python om 2D simulatie modellen snel in productie te brengen in Python. Mesa biedt onder andere functionaliteiten voor modulaire componenten, browser-based GUI en een ingebouwde data-collection tool.
Het is in ontwikkeling gebracht als een Python-vriendelijk alternatief voor NetLogo.

Voordelen:

- Is een Python library (meest gemeenschappelijke taal).
- Toegang tot andere Python data science libraries (bijv. Pandas)
- Performance.
- Makkelijk data op te slaan (ingebouwde data collector).
- Mogelijkheid voor een GUI ingebouwd (webbased).
- Architectuur van Mesa is aanpasbaar.

Nadelen:

- Data visualisatie moet worden geïmplementeerd.

Unity:
Unity is een professionele 2D & 3D game Engine, dit houdt in dat er ingebouwde functionaliteiten zijn zoals een Physics Engine, collision detection etc. De tool komt ook met een ingebouwde text-editor waarin met C# o.a. agents geprogrammeerd kan worden. In unity kun je zowel vanaf een high-level games/simulaties maken als vanaf een low-level.

Voordelen:

- Veel voorgebouwde functionaliteit zoals agent gedragen en een physics engine.
- Kan op een high-level (click and drag) gebruikt worden maar op een low-level (C#).
- Je bouwt je omgeving met een visuele preview.

Nadelen:

- Data visualisatie is niet ingebouwd.
- Hoge learning curve.
- Veel bestaande functionaliteit achter paywalls.
- Duurt relatief lang om een simulatie te starten/stoppen.
- Beperkt met data opslaan en visualiseren.
- Geen simpele ingebouwde GUI naast de tool zelf.

SF Matrix

Beoordeling gebaseerd op Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software.

	NetLogo	Unity	Mesa
Geschiktheid			
Geschiktheid ontwikkelen modules	2	1	3
Efficiëntie, snelheid van simulatie.	1	2	2
Compatibiliteit (externe data)	1	2	3
Haalbaarheid			
Vaardigheden van projectleden	2	1	2
MVP maken in 2 weken.	3	1	3
Totaalscore	**9**	**7**	**13**

Definitieve tool-keuze

In de SF analyse scoort Mesa het hoogst voor zowel geschiktheid als haalbaarheid en is hierdoor de meest geschikte keuze op basis van alleen de score. Vanwege de score voor Unity was snel duidelijk dat dit geen geschikte keuze is. Dit komt voornamelijk door de complexiteit van de tool in combinatie met onze beperkte kennis en tijd. Hoewel NetLogo ook goed scoort in zowel geschiktheid als haalbaarheid, hebben besloten om a.d.h.v. de SF analyse, Mesa als simulatie-tool te gebruiken. Hierin was doorslaggevend dat Mesa niet alleen voordelen heeft van NetLogo zoals een relatief beginners-vriendelijke ervaring, maar het ook mogelijk maakt om andere (bekende) Python libraries te gebruiken. Ook de modulariteit/mogelijkheid om de source-code in te zien is een groot voordeel van Mesa ten opzichte van NetLogo en Unity.

Referenties:

Wikipedia contributors. (2020, 14 december). Nagel–Schreckenberg model. Wikipedia. Geraadpleegd op 10 december 2021, van https://en.wikipedia.org/wiki/Nagel%E2%80%93Schreckenberg_model

CHEN, H.-J. (2001). A VARIED FORM OF THE NAGEL–SCHRECKENBERG MODEL. International Journal of Modern Physics B, 15(26), 3453–3464. doi:10.1142/s0217979201007348

Wikipedia-bijdragers. (2021, 30 april). MoSCoW-methode. Wikipedia. Geraadpleegd op 10 december 2021, van https://nl.wikipedia.org/wiki/MoSCoW-methode

Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review*, *24*, 13-33.