Mesa

MESA ABMS

Jason de Mey, V2B, Simulation, 2-12-21

HOGESCHOOL UTRECHT

Inhoud

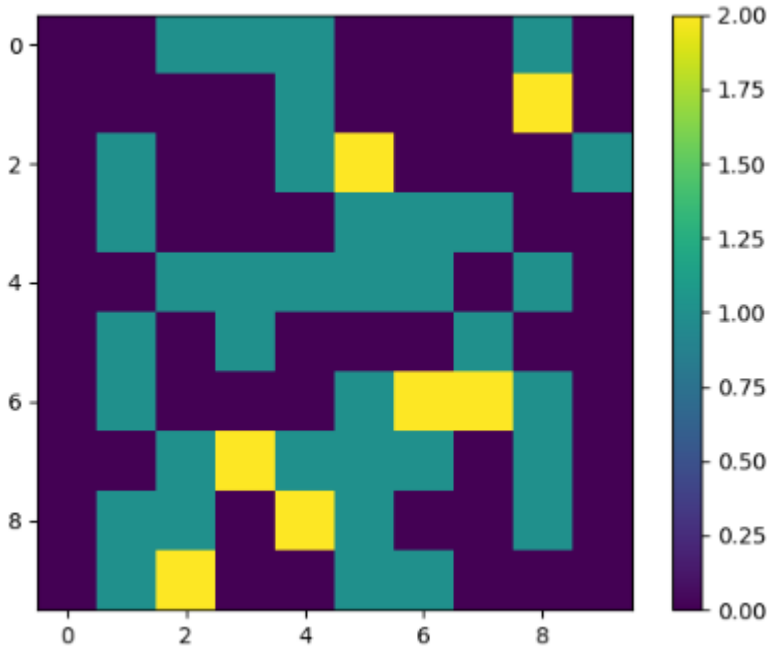
1. Inleiding

Tijdens dit project wordt er een tutorial van Mesa gevolgd, een modulair framework voor het bouwen, analyseren en visualiseren van agent-based modellen. Daarnaast worden de voor en nadelen op een rijtje gezet tegenover zowel Unity als NetLogo.

2. Omschrijving tutorial

De tutorial waarmee wordt gewerkt is als volgt:
https://mesa.readthedocs.io/en/master/tutorials/intro_tutorial.html

In deze tutorial starten alle agents met hetzelfde vermogen, namelijk “1”. Elke stap staat een agent met vermogen van 1 of hoger ook weer “1 unit” af. Wanneer het model door meerdere stappen gaat, zal uiteindelijk te zien zijn dat de perfecte distributie waarbij elke agent een vermogen van 1 had, verandert naar een oneven speelveld. Sommige agents zullen een hoger vermogen hebben dan anderen. Het resultaat van een willekeurige run van het programma is te zien op onderstaande afbeelding.



Afbeelding 2.1: Willekeurig resultaat van de ABMS.

3. Aanpassing aan de tutorial

Agents staan hun geld niet af als ze een andere agent tegenkomen met een vermogen hoger dan 5.

4. Mesa tegenover NetLogo en Unity

Mesa heeft een iets hogere “learning curve” dan NetLogo, tenzij men al gewend is aan programmeren in Python. Zelf vind in NetLogo wel gebruiksvriendelijker, omdat je meteen al een GUI klaar hebt staan en het toevoegen van functies aan buttons of sliders gaat ook gemakkelijk. Verder heeft het programma zelf een simpele omgeving met weinig functionaliteiten. Een nadeel aan NetLogo is dat het een relatief oud programma is, hoewel dit voor simpele simulaties geen probleem hoeft te zijn.

Unity daarentegen heeft de hoogste “learning curve” van de drie, omdat je hiervoor eerst enigszins vaardig moet zijn in C#. Daarnaast heeft de omgeving ontzettend veel opties zoals bijvoorbeeld het veranderen van de camera, het toevoegen van layers etc. Dit is naar mijn mening voor de simpele simulaties binnen de scope van dit project te veel van het goede. Wel zou ik in mijn vrije tijd zonder enige druk van een project graag een hobby project doen in Unity.

Zelf heb ik uiteindelijk gekozen voor de middenweg, Mesa, omdat mijn andere groepsgenoten voor NetLogo kozen en omdat de visualisaties net iets mooier ogen in een Python IDE of notebook, dan in NetLogo. Daarnaast is het fijn werken met een model en een agent superclass die al zijn opgesteld voor je. Ook de ingebouwde “get_neighborhood” en “get_cell_list_contents” zijn handig voor het vinden van mogelijke stappen om te maken en het zien van andere agents op een eigen cell.

In het vervolg zal ik sneller naar de tool NetLogo grijpen, vanwege de simpliciteit. Daarnaast vond ik het bij NetLogo fijn om per stap te kunnen zien wat een agent doet en dit zelfs tijdens het draaien van een programma kan wijzigen met bijvoorbeeld een slider.

5. States

Een agent heeft de mogelijkheid om geld weg te geven en geld te ontvangen.
Ook kan een agent van plek veranderen door naar een andere cell te bewegen, hierbij kan deze gezien worden door een andere agent.

Wanneer agents zich op dezelfde cell bevinden en meer dan 0 units aan geld hebben, wordt dit weggegeven, zolang de andere agent niet een hoger vermogen heeft dan 5.

Dit houdt dus in dat de staat van een agent bevat waar deze zich bevindt op het grid, hoe hoog zijn of haar vermogen is en of er een andere agent op dezelfde cell staat.

6. Dichotomieën

6.1 Accessible VS inaccessible

De waarneembare omgeving voor een agent betreft 9 cells, namelijk degene waar hij of zij zich op bevindt en de 8 omliggende cells. Alle cells daarbuiten kunnen niet gezien worden, dus zou ik beargumenteren dat de omgeving in zijn geheel inaccessible is voor een agent, tenzij we het over diens omgeving ofwel neighborhood hebben.

6.2 Deterministic VS non-deterministic

De acties die de agents uitvoeren hebben allemaal een gegarandeerd effect, namelijk wel of geen geld uitdelen.

6.3 Episodic VS non-episodic

De actie die een agent uitvoert heeft niet veel invloed op de volgende acties, tenzij een agent in een staat verkeert waarbij deze een vermogen heeft van 0. Wanneer een agent een vermogen heeft van 0 heeft dit juist wel invloed op diens volgende actie, namelijk het niet kunnen afstaan van geld. In alle andere gevallen maakt het de agents niet uit waar een ander staat en hoeveel vermogen deze heeft, dus ik zou dit een episodic omgeving noemen.

6.4 Static VS dynamic

De omgeving is statisch, er verandert niks aan het grid zelf, wat zou kunnen worden toegevoegd is dat een agent een cell op het grid willekeurig vies maakt. In dat geval is het een dynamische omgeving.

6.5 Discrete VS continuous

De stappen die het programma aflegt, de staten van de agents en het grid zijn allemaal eindig. Hierdoor zou ik beargumenteren dat het om een discreet systeem gaat.

7. Tegenovergestelde dichotomieen

Het zou precies tegenovergesteld zijn wanneer er agents een willekeurige dankbare of ondankbare reactie geven op het ontvangen van geld. Dit zou betekenen dat, ten opzichte van nu, een actie een non-deterministisch effect kan hebben.

Ook zou het tegenovergesteld zijn, wanneer er een willekeurige staat van het weer zou worden toegevoegd. De wereld is dan dynamisch, ten opzichte van de statische staat van nu. Zolang er niet op gereageerd wordt door de agents, heeft het geen toegevoegde waarde.

Tot slot zou het ook tegenovergesteld zijn, wanneer het grid waarop de agents zich kunnen verplaatsen “oneindig” is. Dit kan misschien gedaan worden met het procedureel genereren van meer terrein, maar ook dit is overbodig. Misschien komen de agents elkaar nooit meer tegen om geld uit te wisselen.