Intro to agents and agent-tools

Vraag 1:

Ik volgde de tutorial van ‘King of the Mesa’. Mesa is een agent-based modeling framework in Python, dus het is geen eigen taal zoals netlogo. Gebruikers kunnen met Mesa snel een agent- based model maken met behulp van ingebouwde componenten, zoals grinds en ‘schedulers’. Je kan de implementaties visualiseren met behulp van een browser gebaseerde interface.

Voordelen:

- Beschikbaar op een van de meest bekendste programmeertalen.
- Het kan samenwerken met vele Python libraries
- Veel informatie op het internet
- Kan Mesa gebruiken in Jupyter Notebook

Nadelen:

- Moeilijker maken van visualisatie

Bij deze opdracht heb ik de tutorial op canvas gevold. In deze opdracht word een simpele vorm van economie gesimuleerd. Er zijn een bepaald aantal agents en elke agent begint meet 1 geld. Bij elke stap van het model verplaatst de agent en geeft de agent, als deze agent geld heeft, 1 geld weg aan een andere agent als deze in de buurt is. Het element dat ik heb toegevoegd is dat een agent, als deze genoeg geld heeft, aan elke agent in dezelfde grid 1 geld geeft. Bijvoorbeeld: op de grid met coördinaten (x=2, y=5) staan 6 agents. Als een agent genoeg geld heeft om alle agents uit die grid geld te geven dat doet deze agent dat. Als een agent dat niet heeft dan geeft de agent 1 geld aan een random agent uit diezelfde grid, maar dit was bij de tutorial al geïmplementeerd.

Vraag 2:

1. De initiele staat is de staat van een agent als deze word aangemaakt.De initiele staat van de agent bestaat uit een random positie op de grid dus een x en y coördinaat, elke agent begint met 1 ‘wealth’ en elke agent krijgt een unieke id.

2. Bij de ‘See’ functie kijkt een agent naar de omgeving. Bij de functie ‘See’ kijkt de agent naar andere agents in dezelfde grid. De perceptie is dan welke agents in dezelfde grid zitten.

3. Deze functie oefent een actie uit op de interne staat. De ‘act’ functie verplaatst de agent naar één van de 8 omringde grids om de agent heen. Vervolgens kiest elke agent één agent uit dezelfde grid en haalt één ‘wealth van zichzelf af en geeft deze aan die agent, tenzij de ‘wealth’ van een agent even groot of groter is dan het aantal agents in dezelfde grid, dan geeft de agent één ‘wealth’ aan elke agent. Als een agent 0 ‘wealth’ heeft dan geeft deze agent geen ‘wealth’ uit.

4. Deze functie update de staat naar een nieuwe staat door de perceptie erbij te voegen. De ‘update’ functie voegt of haalt een aantal van de ‘wealth’ af van een agent en update de positie van de agent.

Vraag 3:

1. Mijn omgeving is Inaccessible, want de agent kan alleen de informatie van de agents die op dezelfde positie staat zien. Dus de agent heeft een beperkt zicht waar het informatie vandaan kan halen.

2. Mijn omgeving is Non-Deterministic, want er zitten wat random keuzes in zoals: de beginpositie van de agent, de gekozen agent om geld aan te geven, de kant waar de agent naartoe beweegt. Ook is de volgorde van welke agent zijn actie uitvoert random.

3. Mijn omgeving is Episodic, want de agent kijkt alleen naar de informatie van de huidige staat en niet naar de toekomst/verleden.

4. Mijn omgeving is Static, want de omgeving word alleen veranderd door de actie van een agent en heeft geen andere processen die de omgeving veranderd.

5. Mijn omgeving is Discrete, want er is een grid waar de agents in rond bewegen. Stel dat mijn simulatie op een open veld had afgespeeld, dan was het continuous geweest.

Vraag 4:

1. Door de omgeving van inaccessible naar accessible te veranderen kan de agent naar alle agents in de grid kijken en dan de agent bijvoorbeeld aan een random agent één ‘wealth’ geven. Mij lijkt deze aanpassing niet nuttig aangezien de grid dan onnodig word en deze manier was de basis van de tutorial. Zo is de simulatie dan eenvoudiger dan als de omgeving inaccessible zou zijn.

2. Discrete naar continuous. Deze aanpassing zorgt ervoor dat er geen grid meer komt maar meer een vrij veld. Hier kunnen agents dan bijvoorbeeld verschillende stapgrootte maken. Zo zou bijvoorbeeld agent 1 0.5 op de x-as kunnen bewegen en agent 2 bijvoorbeeld 3 op de y-as. Zo komt er een meer menselijke simulatie van. Hierdoor zou je dan ook de aanpassing moeten maken in hoever een agent van elkaar af moet staan om ‘wealth’ te kunnen geven/ontvangen. Hiervoor kan je bijvoorbeeld een radius gebruiken. Deze verandering zou best nuttig zijn om de simulatie te verbeteren, want het lijkt dan meer op het echte leven.

3. Non-Deterministic naar deterministic. Hierdoor zouden er dan geen random keuzes meer gemaakt mogen worden. Zo zou iedereen met een bepaalde formule naar een plek op de grid moeten komen. Zo kan agent 1 bijvoorbeeld een naar rechts en 1 omhoog, agent 2 dan 2 naar rechts en 2 omhoog enz. Ook zou de keuze van welke agent nou één ‘wealth’ krijgt voorgeprogrammeerd zijn. Dit kan door bijvoorbeeld de persoon met de minste ‘wealth’ één ‘wealth’ te geven. Maar het enige wat mij onrealistisch lijkt is het programmeren welke volgende stap de agent neemt, dus of de agent een stap naar recht, links, boven of onder beweegt. Hierdoor vindt ik dat deze aanpassing de simulatie niet nuttiger maakt.