Kunstmatige intelligentie binnen de Beergame

Lody Koop
Begeleider: dr. G.A.W. Vreeswijk
7,5 ECTS

14 Juni, 2015

INHOUDSOPGAVE

1

INLEIDING

Het zit in de aard van de mens om de wereld om zich heen te willen begrijpen. Deze schitterende wereld waar we in leven is gelukkig niet zo makkelijk te begrijpen en het is eigenlijk maar de vraag of we deze ooit helemaal zullen doorgronden. Toch zoeken mensen naar antwoorden waarom dingen gebeuren en zo zijn in de complexe wereld om ons heen. Eén van de gereedschappen die we gebruiken om dingen over de wereld te leren zijn modellen. Modellen zijn een simpelere abstractie van de werkelijkheid en dienen er toe om ons nieuwe inzichten te geven over de complexe dingen om ons heen.

1.1 BEERGAME BINNEN HET GEBIED VAN KUNSTMATIGE INTELLIGENTIE

Een extra complexe deelverzameling van modellen, zijn deze waar men tracht menselijk gedrag en intelligentie te modelleren en de interactie tussen deze te bestuderen. Een van deze modellen is de Beergame. De Beergame is een model wat een productieketen simuleert die bestaat uit intelligente agenten. Onderzoek wordt gedaan naar hoe deze agenten redeneren en waarom de keuzes die de agenten doen die lokaal het beste lijken niet voor een optimaal resultaat zorgen in het gehele systeem. De agenten leren van de vorige gebeurtenissen en baseren hierop de komende keuzes. Ze bezitten dus een vorm van kunstmatige intelligentie.

1.2 DYNAMISCH CHAOTISCH SYSTEEM

De agenten in de Beergame gebruiken simpele duidelijke regels om te leren en optimaal te presteren. Ondanks dat de communicatie en redenatie van de agenten niet complex zijn ontstaat er wel erg complex gedrag als de agenten met elkaar gaan communiceren. Het systeem wat ontstaat door deze agenten met elkaar te laten communiceren is dynamisch van aard. Het gedrag van het systeem is niet lineair, dit houdt in dat als er een kleine verandering in de omgeving gebeurt dit tot een onevenredig grote reactie van het systeem resulteert. Het model is echter wel deterministisch dus onder dezelfde omstandigheden zal dit tot exact dezelfde reactie leiden. Maar omdat een kleine

verandering in omstandigheden tot groot uiteenlopende resultaten leidt is het gedrag van het systeem chaotisch (Strogatz, 2014).

1.3 VRAAGSTELLING

Het gedrag van het huidige model van de Beergame geeft inderdaad inzichten in fenomenen van de echte wereld. Het simuleert een productieketen waar de vraag van de klant maar één keer veranderd en hierna constant blijft. Door de manier van redeneren van de agenten leidt dit vaak tot extreem grillig gedrag wat het "Bullwhip-effect" genoemd word. Dit is ook gedrag wat er in de werkelijke wereld voorkomt. Maar hoe zouden de agenten redeneren op een vraag die constant blijft? Zal dit ongunstige gedrag dan ook optreden? Dit is het meest simpele geval en het model geeft hier nog geen antwoord op. Moet de cognitie van de agenten uitgebreid worden of is de huidige manier van redeneren correct en heeft het overeenkomsten met de daadwerkelijke wereld die het probeert te modelleren. De vraagstelling voor deze scriptie luidt dan ook:

> *Het Beergame model is ontwikkeld om een industriële productieketen te simuleren waar een mogelijk "Bullwhip-effect" optreedt bij bepaalde keuzes van zelfstandig handelende agenten. In het Beergame model beschikken agenten over een vorm van intelligentie om een beslissing te maken hoeveel ze willen bestellen. Kan bij het simuleren van de Beergame de kunstmatige intelligentie van de agenten uitgebreid worden voor een constante vraag, met een juiste manier van redeneren?"*

1.4 INDELING SCRIPTIE

De rest van deze scriptie is verder als volgt ingedeeld:

- Hoofdstuk 2 zal verdere motivatie rondom de Beergame uitleggen.
- Hoofdstuk 3 gaat in op de exacte werking en regels van het model. Na het lezen van dit hoofdstuk zal duidelijk zijn hoe het verloop van het spel gaat en wat de mogelijke huidige vormen van redeneren zijn van de agenten in het model.
- Hoofdstuk 4 geeft een kritische invalshoek weer op de huidige werking van het model.
- Hoofdstuk 5 zal eigen gemaakte uitbreidingen op het model bespreken.
- Hoofdstuk 6 geeft de conclusie op de vraagstelling.

- Hoofdstuk 7 is de bibliografie waar geraadpleegde literatuur staat vermeld en waar men de locatie kan vinden van alle besproken Netlogo implementaties.

2

MOTIVATIE

2.1 HET ONTSTAAN VAN DE "BEERGAME"

Begin jaren zestig zijn meerdere hoogleraren aan de MIT Sloan School of Management onderzoek begonnen naar verschillende effecten die optreden in commerciële productieketens. Een productieketen is het hele proces wat er voor zorgt dat een grondstof uiteindelijk wordt omgevormd tot een volwaardig product en bij de klant terecht komt. Dit is een proces wat uit meerdere stappen bestaat en wordt vaak niet gedaan door één bedrijf. Een simpel voorbeeld van een productieketen is: Fabriek ⇒ Distributeur ⇒ Groothandelaar ⇒ Winkelier ⇒ Klant. Het product wordt gevormd in de fabriek en de rest van de productieketen zorgt voor de distributie hiervan. Omdat een productieketen uit meerdere zelfstandig handelende entiteiten bestaat zijn de resultaten niet optimaal en worden er vaak overbodige kosten gemaakt. Om de effecten die optreden in zo'n productieketen te onderzoeken hebben deze hoogleraren van de MIT Sloan School of Management een simulatie spel genaamd "The Beergame" opgesteld wat inzicht heeft geboden in dit onderzoeksveld.

2.2 DE ORIGINELE BEERGAME

"The Beergame" is een spel wat wordt gespeeld door teams die zo min mogelijk kosten moeten maken. Deze spelers krijgen allemaal een rol als leverancier toegewezen in de productieketen. Ze mogen alleen met elkaar communiceren door middel van bestellingen te plaatsen bij de leveranciers achter hun of bestellingen te leveren aan de leveranciers voor hun. De eerste speler in de keten draait een kaart om waar de vraag van de klant op staat. De spelers moeten proberen de vraag van de leverancier of klant voor hun proberen te leveren. Als dit niet lukt maken ze extra kosten. Ook maken ze kosten voor de producten die ze in opslag hebben. Tussen de levering van producten en bestellingen zit in het spel een vertraging.

De vraag van de klant die wordt gegeneerd aan het begin van het spel is altijd van dezelfde typische vorm. In het begin is de vraag constant maar na een paar rondes wordt de vraag eenmaal verhoogd en blijft voor de rest van het spel constant. Ondanks dat de vraag heel het spel constant blijft op één verandering na ontstaan

er aanzienlijke effecten in de grootte van de bestellingen van spelers. Ook de kosten zijn aanzienlijk hoger dan wanneer de spelers rationeel zouden handelen en even veel zouden bestellen als de leverancier voor hen heeft gedaan. De spelers raken vaak gefrustreerd en denken dat hun medespelers het spel niet goed hebben begrepen omdat ze steeds extremere bestellingen krijgen te verwerken en de kosten alsmaar hoger worden.

2.3 "BULLWHIP-EFFECT"

De effecten die optreden zijn dat er steeds extremere bestellingen worden gemaakt bij leveranciers verder in de productieketen. De spelers willen door de verandering in vraag en de vertraging in het ontvangen van goederen graag reserves opbouwen in hun opslag, zodat ze volgende verhoging van de vraag het hoofd kunnen bieden. Omdat elke schakel in de productieketen hier hetzelfde over denkt zal bij elke stap verder in de productieketen ook meer en meer overbodige producten worden besteld. De amplitude van bestellingen worden hierdoor groter. Omdat er op een moment veel te veel besteld is en meer binnen komt dan waar de spelers op hebben gerekend stoppen ze met bestellen omdat ze van hun voorraad meer dan genoeg kunnen leveren. Hierdoor is er niet alleen een amplitude in bestellingen maar oscilleert het ook. Dit effect heet het "Bullwhip-Effect" en laat zien dat al bij een simpele verandering in vraag in een simpele omgeving zoals de beergame onnodig hoge kosten kunnen ontstaan.

Het bullwhip-effect is ook iets wat voorkomt in echte commerciële productieketens (Lee & Padmanabhan & Whang, 1997). Het is daarom ook van grote waarde om dit te onderzoeken omdat het voor extreem hoge onnodige kosten zorgt. Een echte productieketen is een systeem wat uit ontzettend veel onderlinge relaties en entiteiten bestaat. Alle eenheden in zo'n productieketen handelen allemaal autonoom en willen zo handelen wat voor hun het voordeligst is. Deze factoren stimuleren het bullwhip-effect alleen maar meer. Om inzicht te krijgen in hoe het werkt is een echt systeem veelal te complex om te onderzoeken, de beergame leent zich hier echter uitstekend voor omdat het een simpele geïsoleerde omgeving is. Het gedrag van keuzes die spelers maken in het spel is onderzocht en zelfs tot formules omgezet die het aantal bestellingen wat een speler doet simuleert. Ook heeft het veel inzichten gebracht in oorzaken en oplossingen voor het bullwhip-effect. Het gebrek aan communicatie en doorzichtigheid van de productieketen zijn factoren die erg belangrijk zijn bij het voorkomen van negatieve effecten.

2.4 NETLOGO IMPLEMENTATIE

Omdat de originele beergame met echte spelers wordt gespeeld en lang duurt kost het veel tijd om de data om te zetten naar informatie en hier vervolgens van te leren. Netlogo is een multi-agent programmeerbare modellering omgeving. Hierdoor is Netlogo een geschikt gereedschap om de Beergame te implementeren. Om onderzoek te doe naar de Beergame heeft O. Densmore het Beergame model geïmplementeerd in Netlogo (Densmore, 2004). De Netlogo implementatie zorgt er voor dat je makkelijk en snel onder verschillende omstandigheden verschillende effecten kan onderzoeken. Het geeft duidelijke grafieken weer zodat men kan nagaan wanneer het bullwhip-effect optreedt, of wanneer juist niet. Dit interactieve systeem laat mensen geïnteresseerd in het management van productieketens belangrijke inzichten zien die kosten in de echte wereld kunnen verlagen. Het doel van de implementatie is daarom net zoals het model het inzichtelijk maken van verschillende effecten op een productieketen onder verschillende omstandigheden en om te illustreren wat betere onderlinge communicatie kan betekenen. De voorgestelde uitbreidingen die in hoofdstuk vijf aan bod komen zijn ook geïmplementeerd in een eigen Netlogo versie van het Beergame model.

3

WERKING MODEL

Dit hoofdstuk begint met het toelichten van alle stappen die er worden gemaakt in het spel en de regels die hier bij horen. Er zal aandacht worden besteed aan de concrete implementatie van het model in Netlogo. Aan het einde worden de drie oorspronkelijke mechanismes achter de manier van bestellen van leveranciers uitgelegd.

3.1 VERLOOP

Het model simuleert een productieketen die bestaat uit vijf schakels. De Customer die voor de vraag zorgt en de vier leveranciers, namelijk de: Retailer, Wholesaler, Distributor en de Factory. In elke stap doet de klant één bestelling bij de retailer. Alle vier de leveranciers voeren in deze zelfde stap verschillende taken uit:

1. Ze ontvangen een lading van oude bestellingen.
2. Ze ontvangen een bestelling van de leverancier lager in de productieketen.
3. Ze leveren de bestelling aan de leverancier lager in de productieketen.
4. Ze doen zelf een bestelling bij een leverancier hoger in de productieketen, gebaseerd op hun huidige hoeveelheid opgeslagen bier.

Het duurt één tijdseenheid voordat een bestelling wordt ontvangen door een leverancier hoger in de productieketen. Het duurt twee tijdseenheden voordat een bestelling wordt geleverd door de leverancier. Dus in totaal duurt het drie tijdseenheden voordat een bestelling wordt verwerkt en geleverd. De optelsom van alle bestellingen die een leverancier doet wordt door hem zelf bijgehouden en wordt de ”supply line” genoemd.

Het doel voor de leveranciers is om de kosten zo laag mogelijk te houden. Voor elke eenheid die de leverancier in zijn opslag overhoudt wordt $0,50 gerekend. Maar als de leverancier niet aan de vraag kan voldoen wordt de bestelling voor de volgende keer meegenomen en is dit aanzienlijk duurder. Dit heet in het spel een backorder en kost $2,00 per besteleenheid per keer.

3.2 MODEL

Er zijn veel verschillende implementaties van het spel in omloop. Ze variëren van simpele webapplicaties die individueel moeten worden gespeeld, tot een simulatie waar echte spelers moeten deelnemen. Zoals eerder vernoemd is het model wat in deze scriptie wordt gebruikt een Netlogo implementatie, dit omdat het makkelijk aan te passen is en een duidelijke precieze weergave geeft van de originele beergame. In de volgende secties worden dingen uitgelegd die voorkomen in het Netlogo model maar deze eigenschappen behoren terug te komen in elke specifieke implementatie van de Beergame.

3.3 OPSLAG

De opslag van de leverancier heet in het spel Inventory, de inventory van een leverancier kan je in het spel onder de leverancier zelf zien. Als de inventory een negatieve waarde heeft betekent dit dat de waarde die je ziet het aantal backorders zijn.

3.4 ORDERS

De agenten in het spel die voor de leveranciers staan zijn van het turtle soort "Biz". Om de leveranciers zijn altijd 4 grijze patches te vinden. Soms wordt zo'n patch ingevuld met een turtle van het type "Orders", deze is geel als het een positieve waarde heeft en rood als het een negatieve waarde heeft. De orders om de leveranciers heen staan voor verschillende dingen:

- De order linksboven van de leverancier staat voor supply, dit zijn het aantal eenheden die hij geleverd heeft. De levering komt echter wel de volgende stap pas aan bij de leverancier voor hem.
- De order rechtsboven van de leverancier staat voor received, dit zijn het aantal eenheden die hij ontvangen heeft.
- De order linksonder van de leverancier staat voor demand, dit is de vraag die hij krijgt naar eenheden van de leverancier of klant voor hem.
- De order rechtsonder van de leverancier staat voor ordered, dit is de bestelling die hij zelf maakt en doorgeeft aan de leverancier die na hem komt in de productieketen.
- De "factory" leverancier heeft als enige ook nog een extra orders turtle die helemaal rechts staat. Wat de factory wil bestellen wordt de volgende stap gemaakt door de factory en in deze orders turtle gezet. De tijdseenheid daarop ontvangt het van deze orders turtle zijn bestelling in de order rechts boven.

In Figuur 1 hieronder is de leverancier dus de Wholesaler in de productieketen. De acht linksboven staat voor het aantal eenheden wat hij gaat leveren. De zes rechtsboven staat voor het aantal eenheden wat hij ontvangen heeft. De acht rechtsonder staat voor de bestelling die hij zelf plaatst bij de volgende leverancier en de negen linksonder staat voor het aantal eenheden wat net bij hem besteld is. De één helemaal onderin staat voor het aantal eenheden wat hij nog in opslag heeft.

Figuur 1

3.5 GRAFIEKEN

De grafieken die onderaan staan laten duidelijk zien wat de effecten zijn van verschillende keuzes die de eindgebruiker maakt. Aan het einde van elke stap die in het model wordt genomen, worden er in verschillende grafieken data van de afgelopen stap opgeslagen.

De grafiek waar Inventory boven staat laat van alle leveranciers zien hoeveel ze in opslag hadden, of hoeveel backorders ze hadden, op elke stap.

De Orders grafiek laat van elke leverancier zien hoeveel ze hebben bestelt bij de opvolgende leverancier. Ook wordt in deze grafiek het aantal bestellingen van de klant weergegeven. De grafiek genaamd cost laat de kosten van elke leverancier op elke stap zien. Boven de grafiek is nog een extra optie toegevoegd om in plaats hier van de totale kosten voor elke leverancier weer te geven.

Phaseplot is een grafiek die voor een variabele naar keuze een fasediagram laat zien. Vooral in de eerste drie grafieken is duidelijk het bullwhip-effect te zien, en daarom zijn deze ook het interessantst voor het model.

Boven de inventory grafiek staan ook drie displays die de status van het model weergeven. My-ticks geeft het aantal gedane stappen weer. Cost zijn de kosten die alle leveranciers hebben gemaakt de afgelopen stap en AvgCost geeft de gemiddelde kosten van de simulatie tot nu toe weer. Deze laatste is de graadmeter om te kijken hoe goed de leveranciers hebben gepresteerd. De gemiddelde

kosten kunnen worden vergeleken met andere instellingen van het model. Vanzelfsprekend zijn de instellingen met de laagste gemiddelde kosten het meest optimaal.

Onder de grafieken staan de opties OrderFirst en PlayStyle, deze kunnen worden veranderd om de regels van het spel aan te passen en zo verschillende spelvarianten te geven. Om het verhaal niet onnodig complex te maken houden we altijd dezelfde spelvariant aan, waar OrderFirst altijd op ”off” staat, en waar PlayStyle altijd op ”RetailerFirst” staat.

3.6 DEMANDSTYLE

Een functie van het model is de knop waar je ”DemandStyle” kiest. Deze knop geeft verschillende opties weer hoe de klant aan het begin van de keten bestelt. De ”Sine” en ”Square” functie simuleren een vraag die in periodes gaat. Bij deze twee opties kan worden gedacht aan bepaalde jaargetijden waar er meer vraag ontstaat voor sommige producten en in sommige jaargetijden minder. De ”Random” optie levert een volledig willekeurige vraag op. De ”Step” functie houdt de vraag van de klant de eerste vier tijdseenheden constant. Na deze 4 tijdseenheden wordt de vraag van de klant eenmalig verhoogd en blijft daarna tot het einde van de simulatie constant. Deze laatste functie geeft weer dat een simpele verandering van vraag van de klant al voor het bullwhip-effect kan zorgen.

3.7 ORDERSTYLE

De leveranciers moeten proberen hun kosten zo laag mogelijk te houden. Een juiste balans tussen opslag van eenheden is hierbij erg belangrijk. Als ze veel in opslag hebben zal de leverancier minder backorderkosten hebben, maar hij zal wel extra moeten betalen voor de opslag. De leverancier heeft maar één manier om deze balans te beïnvloeden in zijn voordeel, dat is door zelf de juiste bestelling te doen bij de leverancier na hem. Het model biedt drie verschillende manieren van redeneren die kunnen worden gekozen. Ze zijn te kiezen onder de knop OrderStyle. In de komende subsecties staan deze verschillende manieren van bestellen uitgelegd.

Customer

Dit is de meest optimale manier van redeneren. De bestelling die door de leverancier wordt geplaatst is altijd gelijk aan die van de klant helemaal aan het begin van de keten. Hierdoor zullen er nooit backorders optreden en blijft de productieketen stabiel. Dit is weliswaar het meest optimaal maar geeft geen realistische weerspiegeling van kennis die leveranciers tot hun beschikking hebben.

Stockgoal

In deze manier van redeneren probeert de leverancier een vaste hoeveelheid van opslag na te streven. Dit doet hij door een "stockgoal" te nemen. Dit is een doel wat de leverancier stelt en bij elke stap probeert hij zo dicht mogelijk tot dit doel te komen. De formule waarmee dit gebeurt in het model is: $stock0 + order0 - inventory - (pending/3)$. $stock0$ en $order0$ zijn constanten en staan voor de initiële stock waarde en order waarde. $Stock0$ is altijd gelijk aan 12 en $order0$ is altijd gelijk aan 4. Ook $stock0 + order0 = 16$ verandert dus de hele simulatie niet. We kunnen de waarde 16 in dit geval dus zien als stockgoal. Hier wordt de huidige opslag genaamd "inventory" afgetrokken, en de waarde van *pending* gedeeld door 3. *Pending* is de hoeveelheid bestellingen die nog moet binnenkomen. Ter illustratie, als er niks in de opslag zou zijn en er is ook niks besteld dan zijn $pending/3$ en *inventory* dus beide gelijk aan nul. De leverancier zal dan precies 16 eenheden bestellen om aan zijn stockgoal te komen. Deze manier van bestellen is het minst optimaal van alle verschillende manieren om tot een bestelling te komen. De exacte hoeveelheid eenheden die de klant bestelt is echter in de echte wereld ook niet altijd voorhanden zoals bij de "customer" methode wel wordt aangenomen, hierdoor is de stockgoal methode geen slechte methode om de manier van bestellen van een leverancier te simuleren. Omdat de leverancier nooit de vraag die zal komen direct kan weten is de methode om altijd naar een bepaald aantal eenheden in de opslag te streven niet onrealistisch. Met deze manier van redeneren probeert de leverancier reserves te creëren in zijn opslag. Deze manier leidt echter wel tot het bullwhip-effect en heeft hierdoor erg hoge kosten.

Sterman

Een belangrijke boodschap die de simulatie van het model uitdraagt is dat als leveranciers van te voren weten wat de vraag is de kosten aanzienlijk zullen worden verminderd. Dit kan inzichtelijk worden gemaakt door met de zogenaamde Sterman formule te experimenteren en zo zien wat de gevolgen zijn als de leverancier meer informatie heeft.

De Sterman methode is verreweg het meest geavanceerd van alle drie en houdt rekening met informatie die de leverancier heeft en waar hij zijn keuze op kan baseren (Sterman, 1988). Het artikel waarop de methode is gebaseerd is geschreven naar de hand van proeven met de beergame, waar leveranciers werden gespeeld door echte proefpersonen. Sterman probeert in het artikel het bestelgedrag te modelleren in een formule en toont aan dat deze significant overeenkomt met de manier hoe werkelijke personen redeneren als

ze de functie van leverancier toegewezen krijgen. De formule ziet er als volgt uit:

$$\begin{aligned} O_t &= MAX(0, \hat{L}_t + AS_t + ASL_t), \\ \hat{L}_t &= \theta L_{t-1} + (1-\theta)\hat{L}_{t-1}, \quad 0 \leq \theta \leq 1, \\ AS_t &= \alpha_S(S^* - S_t), \\ ASL_t &= \alpha_{SL}(SL^* - SL_t). \end{aligned}$$

O_t staat voor het aantal eenheden wat de leverancier uiteindelijk bestelt en dit is dus wat de hele formule uiteindelijk uit gaat rekenen. $MAX(0, ...)$ zegt alleen maar dat het aantal bestellingen niet kleiner kan zijn dan nul.

$\hat{L}_t$ is de waarde van de voorspelling naar de komende vraag van de klant van de agent. Dit wordt uitgerekend met de waardes van $L_t - 1$ en $\hat{L}_t - 1$. L_{t-1} is de vraag van de vorige stap en $\hat{L}_{t-1}$ is wat er de vorige stap werd berekend voor $\hat{L}_t$. θ is een parameter die de gebruiker van het model zelf kan instellen. θ is een getal wat tussen de 1 en 0 zit of even groot is als deze twee en het bepaalt hoeveel waarde de agent hecht aan de vorige vraag of wat er de vorige stap verwacht werd. Als θ dicht bij de 0 komt zal de agent veel waarde hechten aan wat er de vorige keer verwacht werd. Als de θ dicht bij de 1 zit zal de agent veel waarde hechten aan wat er de vorige keer besteld was door zijn klant.

AS_t is de waarde die wordt uitgerekend door van de begeerde opslag S^* de daadwerkelijke opslag S_t af te trekken. De variabele α_s is ook een getal tussen de 1 en de 0 en geeft aan hoeveel waarde de leverancier hecht aan het verkrijgen van de begeerde opslag S^*.

ASL_t is een waarde die veel op AS_t lijkt maar staat dan niet voor opslag maar voor de "supply-line". Dit zijn het aantal bestellingen die gedaan zijn maar nog niet ontvangen en heet in het spel pending. SL^* staat voor begeerde supply-line en SL_t staat voor de daadwerkelijke waarde pending op die specifieke stap. De variabele α_{SL} is een getal tussen de 1 en de 0 en geeft aan hoeveel waarde leverancier hecht aan het verkrijgen van de begeerde supply-line.

Een variatie op de formule is:

$$O_t = MAX[0, \hat{L}_t + \alpha_S(S' - S_t - \beta SL_t) + \epsilon_t]$$

Hier staat S' voor $S^* + \beta SL_t^*$. In het Netlogo model wordt voor de waarde S' de letter Q gebruikt. β staat voor α_{SL}/α_S. De waarde aan het einde van de formule die ϵ_t heet staat voor de foutwaarde die Gaussische ruis is. Omdat de formule gebaseerd is op een menselijke vorm van redeneren en onderzocht is met statistiek is er sprake van deze foutwaarde. De foutwaarde ϵ_t is echter niet in het oorspronkelijke Netlogo model geïmplementeerd.

De formule is gebaseerd op verankering en aanpassing heuristiek (Tversky & Kahneman, 1974). Als mensen iets moeten schatten hebben

ze de neiging om gebruik te maken van informatie van vorige eventualiteiten, deze informatie is het anker. In de formule kunnen we dit terugvinden in de functie die $\hat{L}_t$ definieert. Er zijn twee ankers te vinden in deze definitie namelijk: $\hat{L}_{t-1}$ en L_{t-1}. Aan welk anker meer waarde wordt gehecht hangt zoals al vermeld van de waarde θ af. Als θ een lage waarde heeft zal de agent veel waarde hechten aan wat hij zelf de vorige keer in heeft geschat en deze elke stap bijwerken. Dit proces kan gezien worden als een vorm van leren.

Stermans formule heeft eigenlijk ook een soort stockgoal wat we kunnen interpreteren als S'. In toevoeging hierop houdt deze formule ook rekening met het de verwachte bestelling $\hat{L}_t$ en verwerkt deze ook alvast in het aantal eenheden wat de leverancier gaat bestellen. De parameters voor β, θ, α en Q zijn allemaal vrij te kiezen in het model en hiermee kan gexperimenteerd worden. β, θ en α zeggen allemaal iets over de waarde die de agent hecht aan verschillende facetten. Het varieëren in deze parameters leidt tot erg uiteenlopend gedrag. Q is een samenvoeging van de begeerde supply-line en begeerde opslag, en ook de hoogte van dit heeft directe uitwerking op de kosten en het gedrag van de leveranciers.

Een belangrijke optie die in het model te kiezen is, is de Visibility keuze. Als deze wordt aangezet terwijl de Sterman manier van redeneren wordt gebruikt door de leveranciers wordt niet de Sterman functie om $\hat{L}_t$ te berekenen gebruikt. Visibility geeft de leverancier de waardevolle informatie over wat de vraag is nu vooraf en $\hat{L}_t$ wordt gelijkgesteld aan deze vraag. Het grote verschil is nu dus dat de leverancier geen schatting maakt van wat de vraag zal zijn maar deze al weet voor hij besteld. Als Visibility aan wordt gezet laat de simulatie zien dat het bullwhip-effect niet optreed. Hierdoor zijn kosten duidelijk verminderd en wordt de waarde van informatie voor leveranciers duidelijk naar voren gebracht.

4

KRITIEK

De Beergame en de daarop gebaseerde implementatie van het model proberen inzicht te geven in het gedrag van leveranciers in een simpele productieketen. Er kunnen verschillende omstandigheden en manieren van redeneren gekozen worden. Hier kunnen verschillende uitingen van complex of juist simpel gedrag uit voort komen. De beergame wil illustreren dat met één simpele verandering in het bestelgedrag van de klant al in de hele productieketen complex en irrationeel gedrag kan ontstaan.

4.1 PHANTOM ORDERING

Een vraag die hierbij opkomt en al in de inleiding naar voren is gekomen is waarom er geen metingen zijn gedaan met het simpelste geval waarbij de vraag van de klant altijd constant blijft. De huidige theorie is dat het bullwhip-effect wordt opgewekt door een verandering in vraag van de klant, wat de balans verstoord in de productieketen. Een aanname die hier uit voort vloeit lijkt te zijn dat bij een constante vraag het bullwhip-effect nooit zal optreden. Het klopt dat als leveranciers rationeel zouden handelen ze constant zouden moeten blijven in hun eigen bestellingen. Als leveranciers echter perfect rationeel zouden handelen zouden ze ook op een eenmalige verhoging van de vraag van de klant zelf eenmalig hun bestelling verhogen en daarna weer constant blijven bestellen. Het lijkt dus de moeite waard om na te gaan of bij een constante vraag wel alle redeneringen van de leveranciers optimaal verlopen en of hun bestellingen constant blijven.

In een recenter artikel van Sterman wordt er onderzoek gedaan naar de Beergame met een constante vraag (Sterman & Dogan, 2005). In ook dit onderzoek namen menselijke proefpersonen de rol in van leveranciers en werd duidelijk vooraf verteld dat de vraag van de klant het hele spel door constant zal blijven. De optimale manier zou dus zijn dat alle leveranciers ook het hele spel door zelf constante bestellingen zouden doen. Het frappante wat hieruit naar voren is gekomen, is dat er maar 9 van de 240 proefpersonen het hele spel door deze rationele keuze bleven volhouden en constant bleven bestellen.

In het artikel wordt verklaard dat bij een constante vraag van de klant, het meest optimale is als ook de begeerde opslag S^* en de begeerde supply line SL^* constant blijven. In de originele formule van Sterman is dat ook zo en in het geïmplementeerde model komt dit naar voren als de parameter Q. Als deze waardes beide constant zouden blijven zou het dus betekenen dat bij een constante vraag er geen bullwhip-effect optreedt en dat de bestellingen van de spelers al snel in een evenwicht zullen terechtkomen waardoor de kosten laag zullen blijven. De uitkomsten van het onderzoek van J.D Sterman en G. Dogan laat echter zien dat deze balans niet altijd ontstaat. De begeerde opslag S^* lijkt met een constante vraag nog steeds constant te blijven maar Sterman en Dogan geloven dat de begeerde supply line SL^* gedurende het spel kan variëren.

Het geïmplementeerde model zou dus completer kunnen worden gemaakt door een constante vraag van de klant te introduceren. Hierbij moet het ook mogelijk zijn dat de leveranciers niet met een constante begeerde supply line SL^* redeneren. Als het artikel van Dogan en Sterman gevolgd zou worden, zou deze manier van redeneren in deze situatie dus een realistischer beeld geven dan de oudere "Sterman-methode" die in het geïmplementeerde model gebruikt is. Dit omdat een groot aantal mensen uit het onderzoek ook niet zo blijkt te redeneren bij een constante vraag. De proefpersonen hebben laten zien dat ze extra onnodige bestellingen doen omdat ze niet zeker zijn van wat de andere leveranciers in de productieketen doen. Deze onnodige bestellingen worden "Phantom orders" genoemd.

4.2 FOUTWAARDE ϵ_t

Een ander punt wat een toevoeging kan zijn op de implementatie van het model is de foutwaarde ϵ_t. Deze foutwaarde wordt wel door Sterman toegevoegd in zijn originele formule maar is niet terug te vinden in de implementatie. Het model is nu deterministisch en geeft bij de zelfde parameters altijd het zelfde resultaat. Dit is ook wenselijk want de doel van het model is het illustreren van wat de consequenties zijn van verschillende omstandigheden. Van een andere kant zou men kunnen zeggen dat het model menselijk gedrag probeert te modelleren. Misschien kan daarom het toevoegen van een toevalsfactor die ook bij werkelijk menselijk gedrag wordt gemeten wel tot extra waardevolle inzichten leiden.

5

UITBREIDINGEN

Uit Hoofdstuk 4 is gebleken dat er nog heel wat verbeteringen kunnen worden toegevoegd. In dit hoofdstuk bespreken we mogelijke oplossingen voor de kritiekpunten die eerder aan bod zijn gekomen en hoe deze geïmplementeerd zijn. Alle uitbreidingen zijn in Netlogo gemaakt maar kunnen ook op andere implementaties van de beergame worden toegevoegd. De secties over constante vraag en Phantom ordering zijn gebaseerd op het het eerder besproken artikel van Sterman en Dogan. De foutwaarde ϵ_t is een eigen voorgestelde uitbreiding. De eigen aangepaste Netlogo implementatie en het origineel van O. Densmore zijn te vinden op: `https://github.com/Lodykoop/Scriptie`

5.1 CONSTANTE VRAAG

Aan de bestaande functies van het geïmplementeerde model die de vraag van de klant simuleren is nu een keuze toegevoegd om de klant constant te laten bestellen. Nu er een constante vraag mogelijk is van 4 eenheden per keer kan er worden gekeken wat de verschillende resultaten zijn van de al bestaande manieren van redeneren die leveranciers gebruiken om te bestellen.

De customer methode laat net als bij de stepfunctie zien dat ook bij een constante vraag de leveranciers constant blijven bestellen. Dit is logisch want het aantal eenheden wat de leverancier bestelt is altijd gelijk aan wat de klant bestelt.

De Sterman methode geeft in het begin een kleine schommeling maar ontwikkelt zich uiteindelijk ook tot een situatie waar alle leveranciers constant bestellen en waarbij de kosten van het spel extreem laag zijn.

Een uitzonderlijke uitkomst is dat als de stockgoal methode gebruikt wordt er zelfs op een constante vraag een bullwhip-effect optreedt. Dit is te verklaren doordat de stockgoal die deze methode stelt niet gelijk is aan de constante vraag. Hierdoor proberen de leveranciers op een te simpele manier tot hun stockgoal te komen waardoor de kosten onnodig hoog worden.

5.2 PHANTOM ORDERING

Zoals al besproken in het Hoofdstuk 4 hebben G. Dogan en J.D Sterman een later artikel geschreven waar er wordt geëxperimenteerd met een constante vraag. Voor hen die de exacte theorie achter het mechanisme willen begrijpen wordt aangeraden dit artikel te lezen. Uit de resultaten in dit artikel blijkt dat veel proefpersonen niet met de originele Sterman methode lijken te redeneren, ze gebruiken een andere manier. De onderstaande theorie om Phantom ordering te bereken zijn ook allemaal aangebracht in de verbeterde eigen Netlogo implementatie. De methode is bijna hetzelfde behalve dat de variabele S' die in het origineel stond voor $S^* + \beta SL_t^*$ nu niet meer wordt gebruikt als constante. De begeerde opslag S^* blijft hetzelfde maar leveranciers gebruiken nu een variërende begeerde supply line SL^*. Hierdoor kan S' dus ook variëren.

SL_t^* wordt berekent met:

$$SL_t^* = EAL_t * DesiredAcquisitionRate_t$$

Hierin kan $DesiredAcquisitionRate_t$ ook wel DAR worden berekend met:

$$DAR = \hat{L}_t + \alpha(S^* - S)$$

Voor $\hat{L}_t$ is in de sectie over de Sterman formule al aangegeven hoe deze moet worden uitgerekend. α is dezelfde als al gebruikt werd bij de originele Sterman methode. S is de opslag en is ook een gegeven waarde. Het enige wat er dus nodig is om DAR uit te rekenen is nog een waarde voor de zogenaamde begeerde opslag S^*. Deze waarde kan in de implementatie worden ingevuld onder DesiredInventory en is een parameter die de eindgebruiker meegeeft aan het model.

EAL_t staat voor $ExpectedAcquisitionRate$ en wordt berekend met de formule:

$$EAL_t = w * NAL + (1 - w) * PAL$$

NAL staat voor $NormalAcquisitionLag$. Dit blijft het hele spel constant en is in het artikel de waarde 4 voor de Distributor, Wholesaler en Retailer en de waarde 3 voor Factory. Maar omdat het in het geïmplementeerde model ook 4 is voor de Factory is het dus voor alle leveranciers heel het spel constant 4. De zogenaamde variabele w is een parameter die de eindgebruiker zelf kan invoeren en is een getal tussen de 0 en de 1. De waarde van w zegt hoeveel waarde de leveranciers hechten aan de $NormalAcquisitionLag$ of hoeveel waarde ze hechten aan de $PerceivedAcquisitionLag$.

PAL staat voor $PerceivedAcquisitionLag$ en wordt berekend met:

$$PAL = MAX(NAL, SL/ShipmentsReceived)$$

Het maximum tussen NAL en de Supply Line SL gedeeld door het aantal eenheden wat ontvangen is staat dus voor PAL. Hier zit de

kern van de Phantom ordering want de leverancier kan dus zelf een andere waarde PAL hebben dan de eigenlijke echte waarde NAL die optimaler zou zijn. Hoeveel waarde hier aan wordt gehecht ligt aan de waarde van de parameter w. De leverancier kan hierdoor in een versterkende cirkel terecht komen waardoor PAL steeds extremere waarden kan gaan aannemen.

Als er wordt geëxperimenteerd met de juiste waarden voor de parameters kan er inderdaad soms een bullwhip-effect optreden. Neem bijvoorbeeld in het nu geïmplementeerde model de waardes: $\alpha = 0,30$, $\beta = 0,08$, $\theta = 0,30$, $w = 0,64$ en $DesiredInventory = 9,9$. Als voor $DesiredInventory$ echter een waarde van 12 wordt gekozen dan blijven de orders bij alle leveranciers constant en zal er geen bullwhip-effect ontstaan.

Naast dat er nu in implementatie de mogelijkheid is om Phantom ordering te gebruiken is er ook de mogelijkheid toegevoegd om alleen de Retailer of alleen de Factory te laten redeneren volgens Phantom-ordering, de rest gebruikt dan de originele Sterman methode.

Na deze toevoegingen is het dus mogelijk gemaakt om ook een realistisch beeld te schetsen van mogelijke effecten op een constante vraag van de klant.

Na de eigen uitbreiding van een constante vraag aan het model en de Phantom-ordering manier van redeneren was er ook de insteek dat deze manier van redeneren misschien ook nuttige inzichten zou verschaffen als het werd gebruikt bij een variërende vraag. Bij een varirende vraag slaat deze manier van redeneren echter door in een veel te krachtige zelf versterkende cirkel. Hierdoor ontstaan er geen representatieve resultaten als deze combinatie van parameters wordt gebruikt.

5.3 FOUTWAARDE ϵ_t

Ook is er de mogelijkheid toegevoegd om in het geïmplementeerde model de foutwaarde toe te voegen die ook aanwezig is in de originele formule. Door de knop 'WhiteNoise' aan te zetten kan er bij de knop SD een standaarddeviatie worden gekozen waarmee ϵ_t wordt toegevoegd volgens een normaal verdeelde witte ruis. Het interessante hiervan is dat er ook bij een constante vraag nu met de originele Sterman formule een bullwhip-effect optreedt. Het patroon is weliswaar veel grilliger dan het patroon zonder deze ruis, maar er ontstaat in de bestellingen van de leveranciers wel duidelijk een patroon dat oscilleert. Ook als de leverancier verder in de productieketen zit is er meer verschil in amplitude van de bestellingen die hij plaatst en de opslag die hij heeft.

6

CONCLUSIE

In deze scriptie is de vraag onderzocht of de kunstmatige intelligentie van de agenten aangepast kan worden op een correcte wijze bij verschillende omstandigheden binnen de Beergame. Het model is uitgebreid naar een situatie met een constante vraag. Het is gebleken dat bij deze constante vraag de kunstmatige intelligentie van de agenten geen goede manier van redeneren hebben. Het originele model reflecteerde bij een constante vraag niet het juiste gedrag. Het model is uitgebreid met een een nieuwe manier van redeneren, Phantom ordering genaamd, en de reactie die het dynamische systeem nu geeft komt meer overeen met de werkelijkheid. Dit heeft tot de inzichten geleid dat zelfs wanneer alle externe factoren zoals in dit geval de vraag van de klant constant blijven een dynamisch systeem als de Beergame extreem gedrag zoals een Bullwhip-effect kan vertonen. Dit wordt mede veroorzaakt omdat intelligente agenten zoals de leveranciers in het spel wel redeneren op een manier die op hun lokale niveau rationeel is, maar omdat de combinatie van deze gedragingen samen tot onvoorziene systeemdynamiek leidt. Verder kunnen we ook concluderen dat agenten niet altijd dezelfde manier van redeneren gebruiken. Als de vraag eenmalig wordt verhoogd is het zo dat agenten meer volgens de originele Sterman wijze zullen redeneren. Als de omstandigheden anders zijn zoals bij een constante vraag verandert ook de manier van redeneren bij de agenten in het model. Een goed model met agenten die kunstmatige intelligentie bezitten is dus een model waar de agenten niet in elke situatie op dezelfde manier redeneren, maar waar ook de cognitie van de agenten adaptief en dynamisch is en zich aanpast aan de omstandigheden. Een agent met kunstmatige intelligentie zou dus niet alleen over een leeralgoritme moeten beschikken wat reageert op directe input van de omgeving maar deze agent zou voor verschillende situaties op verschillende manieren moeten leren. Dit is een inzicht wat niet alleen toepasbaar is op de Beergame. Mogelijk vervolgonderzoek kan gedaan worden naar leeralgoritmes die de juiste leeralgoritmes leren te gebruiken. Dit kan voor zowel de Beergame als voor andere casussen met agenten die over een zekere vorm van kunstmatige intelligentie beschikken.

7

BIBLIOGRAFIE

- Strogatz, S.H.(2014). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. (2de druk). Boulder: Westview Press.
- Lee, H.L. & Padmanabhan, V. & Whang, S. (1997). Information distortion in a supply chain: the bullwhip effect. *Management Science*, 43 (4), 546-558.
- Densmore, O. (2004). The Emergence of Stability in Diverse Supply Chains. *SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems*.
- Sterman, J.D. (1988). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science*, 35 (3), 321-339.
- Tversky, A. & Kahneman D. (1974). Judgment Under Uncertainty: Heuristics and Biases. *Science*, 185, 1124-1131.
- Dogan, G., & Sterman, J. (2005). When Less Leads to More: Phantom Ordering in the Beer Game?. *Proceedings of the 2005 International System Dynamics Conference*, Boston: MA.
- Eigen uitbreiding Netlogo model en orgineel van O. Densmore zijn te vinden op `https://github.com/Lodykoop/Scriptie`