Het uitladen van de containers op een schip naar de kade met gebruik van Reinforcement Learning

Sefa Özmen (19066791), Jesse de Lange (19043856), Eric Maat (20162928), Joanne Pals (20189176), Martti Groenen (19174837) & Ayrton Donderwinkel (22140468)

Minor: Applied Data Science
Groep: 5
Datum: 24-01-2023

Abstract

Wanneer containers op het schip getransporteerd moeten worden naar de kade is het belangrijk een indeling te maken waarin je de kosten en vertragingen van volgende processen minimaliseert. Dit probleem introduceert echter meerdere onzekerheden en beperkingen. De containerindeling aan boord van het schip staat vast en containerkranen zijn niet op elke terminal ten alle tijden bruikbaar. Daarnaast hebben de containers verschillende bestemmingen en zullen ze op de beperkte kaderuimte optimaal gestapeld moeten worden om achtereenvolgende procedures zo vlot mogelijk te laten verlopen. In deze paper presenteren we een Reinforcement Learning model dat op dit gedeelte van het Container Stacking Problem is toegepast. Hierin maken we gebruik van een Deep Q-learning algoritme. We beperken ons tot het gebruik van slechts één reachstacker voor het transport van de containers.

Trefwoorden:
Container Stacking Problem, Reinforcement Learning, Deep Q-Learning, Neuraal Netwerk, Reachstacker

---

1. Inleiding

In 2021 zijn er in de Rotterdamse haven 15,3 miljoen containers aan- en afgevoerd (Centraal Bureau voor de Statistiek, 2022). Al deze containers zijn vanaf het schip op de kade gezet om vervolgens weer op een ander schip te gaan. Er is nog geen concreet antwoord over hoe dit zo vlot mogelijk kan verlopen. Daarom gaan we dit onderzoeken. Het hele probleem aanpakken is erg groot. Daarom richt deze paper zich op de containers vanuit het schip op de kade zetten, rekening houdend met welk schip de container weer weg moet (prioriteit). Hieruit volgt de onderzoeksvraag: Welke methode kan het Container Stacking Problem oplossen met één reachstacker bij het uitladen van de containers?

Lee en Hsu (2007) hebben een Lineair Integer Programmeer model gemaakt om het pre-marshalling probleem op te lossen. Dit probleem betreft containers die worden herplaatst van een deel van de kade naar een andere plek op hetzelfde deel van de kade om zo uiteindelijk een indeling van de kade te krijgen die voldoet aan bepaalde criteria. Met dit model hoeft het schip minder lang aan de kade te liggen. Het model maakt een plan om de containers in de kade te herplaatsen in hetzelfde deel van dezelfde kade om in de toekomst geen containers meer te hoeven herplaatsten bij het inladen van de containers. Het doel van dit model is het minimaliseren van het aantal keer dat een container herplaatst moet worden in dit proces.

Salido et al. (2012) hebben een decision support system gebruikt. Hiermee minimaliseren ze het aantal container herplaatsingen en de tijd van het in- en uitladen van de schepen. Daarnaast houden ze rekening met de prioriteit van de

container. Om tijd te besparen voorspellen ze met een heuristiek het aantal containers die waarschijnlijk herplaatst moet worden. Door gebruik te maken van Greedy Randomized Adaptive Search Procedure, een heuristiek die gebruikt wordt voor gecombineerde optimalisatie problemen, vinden ze een haalbare oplossing. Deze twee methodes hebben zij samengevoegd tot één model.

Een kade kan honderden gebieden hebben waar containers op mogen staan. In deze paper pakken we één gebied van een kade om het klein te houden. Dit gebied heeft vier containers in een rij, vier in een kolom en vijf hoog. Vijf containers hoog omdat dat het maximale aantal op elkaar te zetten container is. Vier containers in een rij en in een kolom zodat er ook containers tussen andere containers neergezet kunnen worden, zodat het meer op de werkelijkheid lijkt. De containers worden door één reachstacker van het schip opgepakt en op de kade neergezet. Deze reachstacker kan alleen de bovenste container van een stapel pakken aan de lange kant van de container zelf. Ook kan hij alleen containers uit de buitenste stapels pakken. We houden aan dat de containers allemaal dezelfde grootte en gewicht hebben. Wanneer het uit te laden schip aankomt is de kade leeg. Als alle containers zijn uitgeladen is de kade vol en wordt de indeling door middel van scorefuncties geëvalueerd.

Het doel van deze paper is het minimaliseren van het aantal herplaatsingen van containers tijdens het inladen van het schip. Hiervoor kijken we naar het aantal containers van de hoogste prioriteit, die als eerste weg moeten, waar de reachstacker niet bij kan zonder een container van een lagere prioriteit te moeten verplaatsen. Om dit te kunnen oplossen maken wij gebruik van Reinforcement Learning.

Reinforcement Learning is een aparte tak van machine learning en valt dus niet onder supervised-, unsupervised- of semi-supervised- learning. Reinforcement Learning gebruikt geen vooraf gedefinieerde dataset bij het trainen. Dit geeft als voordeel dat het gebruikt kan worden om de computer praktische opdrachten te laten leren zonder een aparte dataset te gebruiken. Het wordt dus ook gebruikt bij onder andere het programmeren van robotarmen. Een ander voordeel is ook dat het werkt met een digitaal environment. Hierdoor kunnen de kosten voor het gebruik van fysieke apparaten/processen minimaal gehouden worden tijdens het trainen van het model. Ook kan er bij een Reinforcement Learning model flexibeler worden gewerkt omdat je er geen dataset voor hoeft aan te passen. Je voegt hiervoor een restrictie toe aan het model om het een kant op te sturen (bijna hetzelfde als bij linear programmeren). Reinforcement Learning is ook een praktisch volgbare methode. Omdat het bijna hetzelfde leert als een mens (trial and error en de uitkomsten daarvan onthouden), weet je precies wat de uitkomsten van het model betekenen, en kan je ook makkelijker praktisch beredeneren wat er precies gebeurt.

2. Methodiek

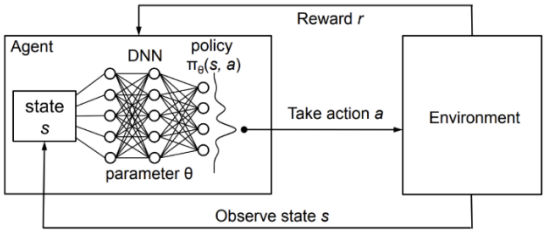
2.1. Reinforcement Learning uitleg

In een Reinforcement Learning model werkt een agent in op een environment. De agent voert bepaalde stappen uit. Ten eerste bekijkt de agent de huidige status van het environment. Met deze observatie selecteert de agent een actie. Hierna geeft het environment door wat voor effect deze actie heeft gehad op het environment. Bij dit effect hoort de *reward*, *new state* en of het spel afgelopen is. De reward geeft aan de agent feedback terug over de kwaliteit van de actie die het heeft uitgevoerd. Door de feedback kan de agent voorspellen welke acties een hogere kans heeft op goede resultaten.

2.2. DQN Uitleg

Het Reinforcement Learning algoritme dat in dit model gebruikt wordt is een Deep Q-Learning algoritme. Dit algoritme gebruikt een Neuraal Netwerk om acties te kiezen. Een ander kenmerk van een Deep Q-Learning algoritme is dat dit een geheugen heeft.

Hiermee onthoudt het algoritme de uitkomsten van alle voorgaande zetten die zijn uitgevoerd. Door dit geheugen kan het algoritme de gewichten van het neurale netwerk steeds aanpassen, waardoor er steeds betere uitkomsten vertoond worden. Voor een voorbeeld van een Deep Q-learning Network Reinforcement Learning model zie figuur 1.



Figuur 1: DQN Reinforcement Learning model

2.3. Toepassing op containerprobleem

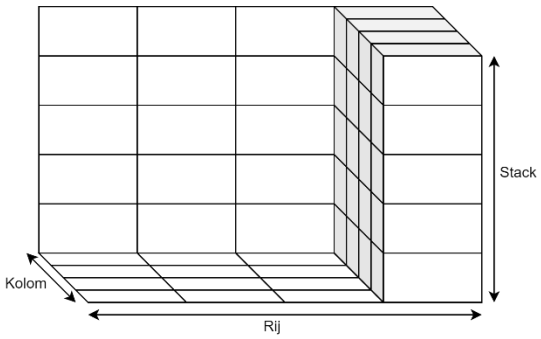
Om deze theorie toe te passen op het Container Stacking Problem moeten er wat delen aangepast worden. Dit betekent dus dat er containers met verschillende prioriteiten ingedeeld worden.

Hierbij hebben we een eigen environment, op maat voor het probleem, gemaakt. Er is gekozen om een driedimensionale matrix te gebruiken om de containers op in te delen. Verder is er ook gekozen om het gebied van de kade op te delen per rij. Hierdoor heeft het neurale netwerk minder kans op slechte moves, en kan hierdoor dus ook efficiënter werken. Bij een kade met vier rijen, vijf kolommen en vijf hoog zouden er dus vier keuzes zijn. In deze rij worden de containers op de achterste stapel ingevuld, wanneer dit vol is, wordt de container in de volgende kolom van deze rij gezet. In figuur 2 wordt gevisualiseerd in welke volgorde dit wordt geplaatst.

Dit environment beloond of straft de agent wanneer er een goede of slechte zet wordt gemaakt. Als de agent een container probeert neer te zetten dan is dat een zet. Bij ons model wordt de agent alleen maar beloond op basis van hoe “goed” een zet is. De agent wordt heftig gestraft als hij een zet probeert te maken in een rij dat al vol is. Het environment berekent hoe goed een zet is door te kijken naar wat voor container de agent in welke rij probeert te zetten. Met de volgende formule wordt er gekeken wat de score van een zet is.

$$Reward = \frac{c_t}{c_{pr}}$$

Hierbij is c_t het totaal aantal containers in de rij waar de container is geplaatst en c_{pr} is het aantal containers van dezelfde prioriteit in dezelfde rij. Uiteindelijk geeft dit per zet een reward tussen de 0 en de 1. Deze reward zorgt ervoor dat de containers worden geclusterd.



Figuur 2: Visuele representatie van een gebied van de kade van vier containers in een kolom, vier in een rij en vijf hoog.

Bij dit environment is er ook een neuraal netwerk nodig om de beslissingen te maken. In de volgende formule is te zien hoeveel input nodes ons neuraal netwerk heeft.

$$N_i = p * r + 1$$

Hier is N_i het aantal input nodes, dus het aantal waardes dat het neurale netwerk als input heeft. p is het aantal verschillende prioriteiten op de kade en r is het aantal rijen in de kade. Hierdoor krijgt het neuraal netwerk per rij mee hoeveel van welke containerprioriteit er in zitten, dit heet een state. Om de prioriteit van de container, die neergezet moet worden, mee te geven aan het model wordt N_i opgehoogd met één (+ 1). Er is bijvoorbeeld een kade met twee verschillende prioriteiten en drie rijen, dan

$N_i = 2 * 3 + 1 = 7$.

Ons neuraal netwerk heeft twee verborgen lagen met beide 64 nodes. Verder heeft de laatste laag, de output laag, even veel nodes als dat er rijen zijn in het environment. Hierbij kijkt het neuraal netwerk welke node er uiteindelijk een hogere waarde teruggeeft, dit is de actiewaarde die meegegeven wordt aan het environment. Deze actiewaarde komt overeen met de rij waarin de nieuwe container wordt neergezet.

3. Resultaten

Voor de resultaten zijn er twee score metrieken gebruikt om zijn twee situaties te vergeleken, een random choice en het Reinforcement Learning model. Beide situaties hebben dezelfde willekeurige input. Hierbij is het schip willekeurig ingedeeld, en er wordt telkens in dezelfde willekeurige volgorde containers uitgeladen.

De random choice is een random indeling van de kade, zonder dat er rekening gehouden wordt met prioriteit van de containers. De random choice houdt wel rekening met de fysieke restricties. Hierdoor kan een container niet in de lucht geplaatst worden en kan een container niet op een plek gezet worden waar al een andere container staat.

3.1. Aantal ingebouwde containers

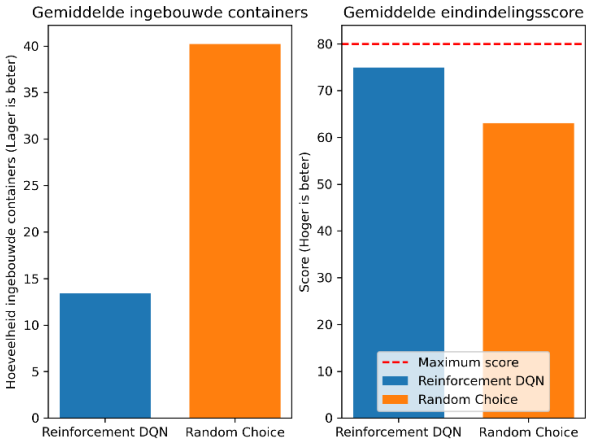
De eerste metriek die we hebben gebruikt is gebaseerd op het gemiddeld aantal ingebouwde containers. Deze berekent per container in een indeling of deze uit de indeling gehaald kan worden zonder dat er een andere container van een lagere prioriteit verplaatst moet worden. Een container waarbij dit het geval is wordt beschouwd als ingebouwd. Hierbij is een lagere uitkomst beter.

Uit het linker staafdiagram in figuur 3 blijkt dat het Reinforcement Learning model indelingen produceert die gemiddeld 13 ingebouwde containers bevatten. Dit betekent dat gemiddeld 83,75 procent van de containers zonder herplaatsingen uit de indeling te halen zijn. Wanneer de containers willekeurig worden ingedeeld ontstaan er indelingen die gemiddeld 40 ingebouwde containers heeft en dus gemiddeld 50 procent zonder herplaatsingen uit de indeling te halen zijn.

3.2. Rij score

De tweede metriek die we hebben gebruikt is gebaseerd op de indelingsscore. Wanneer een container van de kade wordt weggehaald, zijn er twee mogelijke routes. Deze twee routes gaan vanaf de twee lange kanten van de container direct naar de buitenkant van de rij. Deze twee restricties zijn er omdat de reachstacker containers alleen vanaf de lange kanten kan oppakken. Per container op de kade wordt er gekeken hoeveel containers er met een lagere prioriteit op beide routes staan en hoeveel containers in totaal per route. Door het geheel (totaal aantal containers) met het deel (aantal containers met een lagere prioriteit) te delen wordt er een floatwaarde gegenereerd tussen de 0 en 1. Hierbij betekent een waarde dicht bij 0 een heel moeilijke route is en een waarde van 1 een heel makkelijke route is. De score van de container is de hoogste floatwaarde van beide routes. De totale score van de kade is de som van alle scores van de containers op de kade. Hoe hoger de totale score, hoe beter.

Aan het rechter staafdiagram in figuur 3 is te zien dat de door-het-model-gemaakte indelingen een gemiddelde score van ongeveer 75 krijgen en willekeurig gevulde indelingen gemiddeld ongeveer 63 scoren.



Figuur 3: Twee metrieken om Reinforcement DQN met willekeurige keuze te vergelijken.

4. Discussie

In figuur 3 is te zien dat het DQN Reinforcement Learning model gemiddeld veel minder ingebouwde containers heeft bij de indeling dan de random choice (het verschil is 27 containers). Random choice is in dit geval een lat die gelegd wordt om te kijken of het toegepaste model wel degelijk beter presteert. Zoals te zien is, heeft de random choice een duidelijk slechter resultaat als je kijkt naar het aantal ingebouwde containers. Het Reinforcement Learning model heeft het voor elkaar gekregen de containers zo in te delen, dat de containers op basis van de prioriteit voor gemiddeld 83,75% zonder extra verplaatsingen van de kade af te halen zijn, waar een random choice 50% haalt. Dit verschil is te verklaren doordat het Reinforcement Learning model wel rekening houdt met de prioriteit van de containers en hier ook op getraind is doormiddel van het DQN algoritme, de random choice daarentegen heeft geen speciale training of algoritme waar het gebruik van maakt.

De tweede metriek bekijkt het geheel van een ander perspectief. Deze kijkt niet zozeer of de containers ingebouwd zijn, maar hoe toegankelijk de betreffende containers geplaatst zijn. Hier zien we dat het Reinforcement Learning model met DQN algoritme beter werkt dan de random choice. Het Reinforcement Learning model zorgt er hier voor dat 93,75% van de containers direct gepakt kan worden als deze aan de beurt is, bij random choice is dit 78,75%.

Reinforcement Learning heeft dus betere scores dan de random choice methode. Wij adviseren daarom om een Reinforcement Learning model te kiezen voor een betere verdeling dan het willekeurig plaatsen van containers op de kade.

5. Conclusie

In deze paper werd onderzocht hoe een indeling gemaakt kon worden waarin het aantal herplaatsingen van containers tijdens het inladen van een schip kan worden geminimaliseerd. Hierbij wordt er gebruik gemaakt van een reachstacker, die containers transporteert van het schip naar een kade van vier rijen, vier kolommen en vijf hoog.

Om het Container Stacking Problem aan te pakken is gekozen voor het toepassen van Reinforcement Learning. Hierbij is gebruik gemaakt van het Deep Q-Learning algoritme.

Het Deep Q-Learning algoritme maakt gebruik van een Neuraal Netwerk om de beloningen van acties te schatten. Hierdoor kan het leren om de beloningen in complexe omgevingen met veel mogelijke acties te maximaliseren.

Het gebruikte model beloont de agent wanneer containers worden neergezet bij containers van zijn eigen prioriteit. Hierdoor wordt er gezorgd dat het model de containers zo goed mogelijk probeert te clusteren.

In figuur 3 is te zien dat een Deep Q-Learning algoritme te gebruiken is om het aantal herplaatsingen van containers tijdens het inladen vanaf het schip te minimaliseren. Hier is ook te zien dat dit daadwerkelijk beter presteert dan een willekeurige indeling.

6. Toekomstig werk

Bij ons model weet de agent niet hoeveel containers er per prioriteit op het schip aanwezig zijn. Bij een vervolgonderzoek kan dit meegegeven worden aan het model als

input wat waarschijnlijk tot betere resultaten gaat leiden. Daarnaast krijgt de agent nu steeds één container aangereikt die hij op de kade moet neerzetten. Er moet onderzocht worden of dit beter gaat als de agent keuze heeft uit meerdere containers.

Wij hebben één reachstacker gebruikt, maar in het vervolgonderzoek kan er onderzoek gedaan worden naar andere vervoersmiddelen om de containers van het schip op de kade te zetten, of meerdere van dezelfde vervoersmiddelen, bijvoorbeeld een kraan. De reachstacker kan in dit model de containers vanuit één kant oppakken en neerzetten. Het is dus mogelijk om dit uit te breiden, zodat de containers ook van beide lange kanten opgepakt kunnen worden.

In de praktijk is de kade van de Rotterdamse haven niet leeg, daardoor kan het lonen om het model ook te laten trainen op een kade die vanaf het begin al gedeeltelijk gevuld is. Ook is het realistischer te werken met een kade waar meer containers op kunnen staan dan dat er ingeladen worden, waardoor er lege plekken overblijven. Hier moet ook in een vervolgonderzoek aan gewerkt worden.

Literatuurlijst

Centraal Bureau voor de Statistiek. (2022, 14 juli). *Overslag Nederlandse zeehavens herstelt zich in 2021*. https://www.cbs.nl/nl-nl/nieuws/2022/28/overslag-nederlandse-zeehavens-herstelt-zich-in-2021

Lee, Y., & Hsu, N. Y. (2007). An optimization model for the container pre-marshalling problem. *Computers & Operations Research, 34(*11), 3295–3313. https://doi.org/10.1016/j.cor.2005.12.006

Salido, M. A., Rodriguez-Molins, M., & Barber, F. (2012). A decision support system for managing combinatorial problems in container terminals. *Knowledge-Based Systems, 29*, 63–74. https://doi.org/10.1016/j.knosys.2011.06.021