Optimalisatie van het indelen van containers in de haven door middel van Reinforcement Learning en Lineair Programmeren

Milan van Oeveren (22135073), Jurriaan van der Mee (14140373), Ilias Hazali (16070690), Brian Wolvers(19082754), Bram ten Cate (19143451) en Anass Chari (17037190)

De Haagse Hogeschool, Den Haag, 2521 EN, Johanna Westerdijkplein 75.

Abstract

Dit onderzoek is uitgevoerd voor Cofano Software Solutions. Voor containeroverslag in de haven is efficiënt gebruik van kaderuimte van belang voor het optimaliseren van zowel overslagtijd als kosten. Op dit moment wordt geleund op de ervaring van medewerkers om dit te optimaliseren. Het doel van dit onderzoek is om uit te zoeken of data science technieken kunnen bijdragen aan een optimalere overslagtijd en lagere kosten.

We hebben de uitvoeringstijd kunnen optimaliseren door een Reinforcement Learning (RL) model en een Lineair programmeer (LP) model te ontwikkelen. De resultaten tonen dat het RL-model met 1000 games relatief snel traint en na ongeveer twee minuten een ideale optimalisatie vindt, zelfs bij een opstelling van 5x4x4 (lengte, breedte, hoogte). Dit is een te kleine kade en dus niet realistisch. Na het trainen worden toekomstige voorspellingen binnen een fractie van een seconde gedaan. De uitvoeringstijd van het LP-model is ook erg snel, met een gemiddelde rekentijd van 4 seconden op de 3x4x4 opstelling.

Helaas is het zo dat het LP-model 45% van de gevallen een oplossing uit het infeasible gebied weergeeft, waardoor het gebruik ervan minder aantrekkelijk wordt. De resultaten van het onderzoek naar de mogelijkheid en het effect van het gebruik van data science om containeroverslagprocessen in havens te optimaliseren, tonen daarom aan dat RL-modellen consistenter zijn dan de huidige LP-modellen.

Keywords: Reinforcement Learning · Positioneringsproces · Lineair programmeren· Containeroverslag · Optimalisatie

1 Introductie

1.1 Aanleiding

Dit onderzoek is uitgevoerd voor het bedrijf Cofano software Solutions. Dit bedrijf houdt zich bezig met het inzicht geven in logistieke processen en maakt deze volledig beheersbaar door de hele keten te digitaliseren en te optimaliseren. (Cofano, sd)

Eén van de processen die het bedrijf wil optimaliseren is het in- en uitlaadproces van de containers, welke zij in opdracht van de havens waarmee ze samenwerken uitvoeren. De te maken beslissingen tijdens dit proces worden bepaald door de werknemers van de haven en zijn meestal gebaseerd op hun eerdere ervaringen. Deze autonome manier van werken verloopt niet altijd soepel; soms kan het zijn dat een container die bestemd is voor een zeeschip dat veel later vertrekt bovenaan komt te liggen, terwijl een container die eerder ingeladen moet worden onderaan of tussen meerdere containers ligt. Het bedrijf denkt tijdwinst te kunnen genereren door het proces te optimaliseren en vraagt zich af of dat mogelijk is door het toepassen van data science.

Het proces kan onderverdeeld worden in de volgende deelprocessen:

- De volgorde bepalen waarop een kraan de containers van een binnenvaartschip moet uitladen
- Het verplaatsen van de containers naar de juiste plek op de havenkade
- De containers op een goede manier indelen, zodat het inladen op de zeeschepen sneller verloopt
- Het inladen van de containers op de zeeschepen

1.2 Doelstelling

Tijdens dit onderzoek is er focus gelegd op het plaatsen van de containers van een binnenvaartschip op één van de terminals op de haven, uitgaand van een vooraf bekende volgorde van de binnenkomende schepen en containers.

Het doel is om telkens een indeling te vinden voor de containers, zodanig dat het inladen op de zeeschepen sneller kan verlopen.

Om tot een oplossing te komen voor het probleem is er een hoofdvraag opgesteld, welke als volgt luidt: “In hoeverre zijn Reinforcement Learning en Lineair Programmeren geschikt voor het optimaliseren van het stapelingsproces in de haven?”. Deze vraag is geformuleerd aan de hand van het interview met J.Vuurens en wordt beantwoord door middel van literatuur- en veldonderzoek. Het literatuuronderzoek is gebruikt om bestaande informatie te verzamelen over dit type probleemstelling. Vervolgens is met de gevonden informatie een veldonderzoek gestart, waarbij een LP-model en een RL-model zijn ontwikkeld. Binnen deze tools is getracht de situatie zo nauwkeurig mogelijk na te bootsen en deze verder uit te werken tot deeloplossingen.

1.3 Structuurbeschrijving

De paper begint haar opbouw door het uitleggen van waar de gebruikte data vandaan komt, en welke technieken toegepast kunnen worden om het probleem aan te pakken in hoofdstuk 2. In hoofdstuk 3 worden de technieken uitgelegd die zijn toegepast. Vervolgens worden in hoofdstuk 4 de resultaten van het onderzoek beschreven, waarna in hoofdstuk 5 de verbeterpunten van het onderzoek worden besproken. Hierop volgend wordt in hoofdstuk 6 de gestelde conclusie beargumenteerd. Verder worden onder hoofdstuk 7 ideeën uiteengezet die men kan gebruiken, indien voor verdere uitbreiding gekozen wordt. Als laatst is een begrippenlijst opgenomen en worden de referenties benoemd die gebruikt zijn tijdens dit onderzoek.

2 Data

Voor het onderzoek heeft Cofano zeven CSV-bestanden verschaft, waarin uiteenlopende historische data te vinden is. Deze data gaan onder andere over de verschillende machines in de haven, de verschillende containers die verplaatst zijn en welke handelingen er uitgevoerd zijn. Naast deze bestanden is een plattegrond gedeeld waarop de indeling van de kade te zien is.

Aan de hand van onderzoek naar bruikbare methodes, op basis van de bekende informatie, is gekozen voor twee verschillende methodes. De eerste van deze methodes heet Reinforcement Learning. Uit het onderzoek van Kim et al. (2020) blijkt dat Reinforcement Learning een geschikte methode is om verplaatsingen van containers op de kade te minimaliseren. Op de kade worden momenteel keuzes gemaakt op basis van de ervaring van medewerkers en simpele beslisregels, maar door wijzigingen in schema's van aankomst van schepen kan de situatie plotseling veranderen. Het is niet bekend hoe efficiënt de huidige methode is waarop de containers momenteel geplaatst worden op de kade, mede door deze mate van onzekerheid. In dit onderzoek willen wij achterhalen of Reinforcement Learning een uitkomst kan bieden, aangezien het toepassen van Reinforcement learning volgens het onderzoek van Verma et al. (2019) voor een gelijkend probleem met containers tot betere resultaten leidt dan standaard metaheuristieken. Na het trainen van het model kan het binnen maximaal enkele seconden de situatie aanpassen op basis van nieuwe informatie. Reinforcement Learning kan de gekozen beslissingen uitwerken in een environment die de haven nabootst.

Er is voor Reinforcement Learning gekozen aangezien deze techniek mogelijk tot betere beslissingen kan leiden dan die van de havenmedewerkers (Wu, 2021). Mocht de volgorde waarop schepen bij de kade komen veranderen, kan dit snel worden doorgerekend. Daarnaast is er een omgeving bekend, maar ontbreekt een methode om deze te analyseren (GeeksForGeeks, 2022). Binnen dit probleemdomein is de kade als deze omgeving beschouwd. Het RL-model kan op basis van de kade beslissingen nemen.

Voor het RL-model is geen gebruik gemaakt van de meegeleverde data van Cofano. Wel kan deze data van Cofano gebruikt worden voor het valideren van het model wanneer deze uitgewerkt is op de volledige kade. Het bevat bijvoorbeeld data wanneer een container op een bepaalde plaats wordt neergezet, data over die plaats zelf en of een container zich op de kade bevindt of niet. Verder beschrijft het alle acties die uitgevoerd zijn of nog uitgevoerd moeten worden op alle containers. Echter is dit niet geschikt voor het trainen van het model. In de verkregen data ontbreekt naar welk schip een bepaalde container toe gaat of wat de verwachte aankomsttijd van dit schip is. Daarnaast bevat de dataset gelimiteerde historische data, terwijl er op grote hoeveelheden unieke situaties moeten worden getraind.

Voor het testen is er data gegenereerd waarin aangegeven kan worden hoeveel containers er zijn en over hoeveel schepen deze verdeeld moeten worden. Vervolgens worden de containers dan op een willekeurige volgorde aan de verschillende schepen gekoppeld. Elke volgorde van containers voor een bepaald schip komt maar één keer voor in de dataset. Op deze manier is een dataset gecreëerd welke bruikbaar is. Over de container is er bekend van welke klant deze container afkomt en naar welke deze toe moet, maar niet een identificatie van welk schip deze weer op gaat.

Naast Reinforcement Learning is er ook gewerkt aan een lineair programmeer model. De reden dat wij hiervoor hebben gekozen is dat een LP-model aan de hand van de meegegeven restricties de optimale oplossing kan vinden voor een bepaald scenario. Daarnaast zijn er al onderzoeken gedaan die aangeven dat het gebruik van lineair programmeer modellen op havens tijdswinst oplevert. Er is bijvoorbeeld een onderzoek bekend dat eerst een model heeft opgesteld om het containerblok op de haven aan te wijzen, en vervolgens met een ander LP-model bepaalt hoeveel containers voor welk schip er op dat blok moeten komen (Zhang et al., 2003).

Net als bij het RL-model is er geen gebruik gemaakt van de bijgeleverde data van Cofano, maar is er een eigen script geschreven welke een verdeling maakt van een bepaalde hoeveelheid containers over een bepaalde hoeveelheid schepen. Er wordt hierbij een unieke combinatie en volgorde van containers voor de gekozen hoeveelheid schepen random gegenereerd, en als deze combinatie nog niet in de dataset is opgenomen wordt deze toegevoegd. Dit geeft ook de mogelijkheid de gesimuleerde dataset te splitsen voor het valideren van het model. Hierbij gebruikt het LP-model maar één verdeling, en van deze verdeling wordt aan de hand van de restricties de ideale indeling weergegeven.

3 Technieken

Reinforcement Learning

Environment

In het environment van het Reinforcement Learning model wordt de reële situatie nagebootst. De environment bevat een driedimensionaal grid van containers. Voor de X, Y, Z coördinaten is gekozen om de X-as parallel te laten lopen aan de kade, de Y coördinaat loodrecht en horizontaal hierop, de Z coördinaat is de hoogte. Als er een luchtfoto gemaakt wordt en de kade onder in de foto horizontaal loopt. Dan loopt de X-as van links naar rechts, de Y-as van boven naar onder (dus loopt op in de richting naar de kade). De Z-as loopt van de grond de lucht in. Deze aanpak is in lijn met de aanpak van het onderzoek van Verma et al. (2019). De lange kanten van de containers staan gericht richting de kade.

Ook bevat het environment een lijst met containers die geplaatst moeten worden binnen de grid. Elke container heeft een variabele met het id van het schip waar de container uiteindelijk heen moet. De aanname is gemaakt dat de containers aan beide kanten aan de lange zijde verplaatsbaar zijn.

Een Reinforcement Learning environment moet minimaal een aantal onderdelen bevatten om door de agent gebruikt te kunnen worden (Kanade, 2022):

- **State**: in de state wordt de huidige situatie van het environment beschreven. De state is volledig binair, zodat de trainsnelheid van het model optimaal is. De state bevat eerst voor elke plek op de kade of daar een container geplaatst mag worden volgens de opgestelde regels. Zo kunnen containers maar tot een bepaalde hoogte worden gestapeld en een container kan niet tussen de twee lange zijdes van andere containers geplaatst worden. Vervolgens bevat het voor elke X, Y, Z coördinaat of er een container staat of niet. Als laatst beschrijft het voor elke X, Y, Z coördinaat of het schip-id van de volgende container die geplaatst moet worden gelijk is aan het schip-id van de container op de X, Y, Z coördinaat.
- **Reward/Penalty**: Het doel van het trainen van een Reinforcement Learning model is om een zo hoog mogelijke score te halen per game (Verma et al, 2019). In een game wordt voor elke plaatsing van een container een reward of penalty gegeven. Tijdens het berekenen van de reward score wordt er gekeken naar een aantal verschillende onderdelen. Eerst wordt er voor elke container onder de geplaatste container gekeken of deze hetzelfde schip-id heeft. Mocht dit het geval zijn, wordt er een reward gegeven en anders een penalty. Vervolgens wordt hetzelfde gedaan voor alle containers in dezelfde rij, dus dezelfde X waarde. Hierna wordt er gekeken of de geplaatste container er niet voor zorgt dat bepaalde coördinaten in de grid, die niet volgeplaatst zijn, niet meer bereikbaar zijn. Mocht dit wel zo zijn, wordt er ook een penalty gegeven. Als laatst wordt er gekeken of de geplaatste container wel mogelijk zou zijn in de reële situatie. Als dit niet zo is wordt hier ook een penalty voor gegeven. Er is gekozen voor deze rewards, omdat het erg efficiënt is om containers voor hetzelfde schip in dezelfde rij te zetten. Op basis van deze rewards zullen ook zo veel mogelijk containers voor een schip direct bereikbaar zijn, zonder dat daar eerst andere containers voor verplaatst moeten worden. Dit kost namelijk meer tijd en zorgt ervoor dat het proces minder efficiënt is.
- **Action**: de action geeft aan wat de volgende actie is binnen het environment. De mogelijke actions zijn alle X, Y coördinaten. De environment zorgt er zelf voor dat de container binnen de grid op de juiste Y waarde wordt gezet. In het onderzoek van Kim et al (2020) worden de actions up, down, left en right gebruikt, maar om ons model simpel te houden is er gekozen voor het direct plaatsen van de containers op de gekozen locatie. Op deze manier hoeft er nog geen rekening gehouden te worden met trucks.

Validatie

Om het model te optimaliseren, is het model meerdere keren met voor veel verschillende parameters gerund, om zo te kunnen zien welke parameters optimaal zijn voor het model. Er is gekeken naar de invloed van de learning rate en de batch size op het trainen. Voor de learning rates is er gekeken naar een bereik van 0.0003 – 0.03. Deze waardes zijn gekozen door eerdere toetsingen, waarbij te zien was dat een hogere learning rate ervoor zorgt dat het model niet goed traint. Voor de batch sizes is er gekeken naar een bereik van 8 – 128. Een hogere batch size zorgt ervoor dat de tijd per game hoger wordt, waarom er ook niet gekeken is naar extreem hoge batch sizes, omdat dit de trainingstijd zodanig beïnvloed dat het inefficiënt wordt.

Om te controleren of het model niet overfit is, is er een dataset opgesteld met verschillende volgordes van containers. Deze dataset is gesplitst in een trainset en een validatieset. Eerst is het model getraind op basis van de trainset en daarna is de gradient in het model op 0 gezet, zodat het model geen nieuwe dingen leert. Vervolgens is er gekeken of het model goed generaliseert op nieuwe data. Er is gekeken of

het model op nieuwe testdata hetzelfde reageert als op de trainingsdata. Als het model nog steeds een hoge score haalt na het trainen, generaliseert het model goed.

Lineair programmeren

Aannames

Het containerdepot is hetzelfde opgesteld als in de environment van het Reinforcement Learning model en kan dus gezien worden als een driedimensionaal grid. Voor de X,Y,Z coördinaten is gekozen om de X-as parallel te laten lopen aan de kade, De Y coördinaat loodrecht en horizontaal hierop, de Z coördinaat is de hoogte. Als er een luchtfoto gemaakt wordt en de kade onder in de foto horizontaal loopt. Dan loopt de X-as van links naar rechts, de Y-as van boven naar onder (dus loopt op in de richting naar de kade). De Z-as loopt van de grond de lucht in. De lange kanten van de containers staan gericht richting de kade.

Voor het lineair programmeer model (LP-model) is er net als bij het RL-model aangenomen dat containers in een vaste tijdsvolgorde aankomen. Er is bij de depots waar het model op werkt één havenkant waar de schepen voor zowel het in- als het uitladen aankomen. Het levert dan tijdswinst op, om de containers op te stapelen vanaf achteren richting de kade. En altijd tot maximale hoogte te stapelen. Zo hoeven er nooit trucks om het depot heen te rijden, omdat er aan de voorkant geen containers staan. Of containers weg te halen zodat achter die container hoger gestapeld kan worden.

Het uiteindelijke doel van het LP-model is om de haven zo op te vullen dat op elke plek of één container staat, of bereikbaar is voor nieuwe containers. Dit zodat vervolgens bij het opladen van aankomende schepen nooit containers verplaatst hoeven te worden om de containers voor het desbetreffende schip te bereiken. Om dit te bereiken zit het model als volgt in elkaar.

Beginstap

Als eerste wordt naar het aantal containers gekeken dat voor elk schip aankomt. Mocht het zo zijn dat er volledige rijen op de Y-as te vullen zijn met containers die voor één vertrek schip bedoeld zijn, wordt er een rij met een zo hoog mogelijk X coördinaat gevuld met deze containers. Deze wordt gevuld met de containers die volgens de tijdsvolgorde als eerste binnenkomen. De geplaatste containers worden uit een lijst met nog te plaatsen containers gehaald, en de rest wordt volgens het onderstaande model ingevuld.

Sets:

- i = container in tijdsvolgorde van uitladen, i = {1,2,…,I}
- x,y,z = containerplekken in het depot, x = {1,2,…,X}, y = {1,2,…,Y}, z = {1,2,…,Z}
- v = schip (vessel) waarop de container ingeladen wordt, v ={1,2,…,V}

Parameters:

- $LD_{i,v}$ = 1 als container i voor schip v bestemd is
- $Travel_x$ = reistijd van in-, uitlaadpunt naar de x coordinaat.

Variabelen:

Er zijn voor dit model 2 variabelen gebruikt, beide binair

- $C_{i,x,y,z}$ = 1 als container i wordt op plek x,y,z geplaatst
- $E_{i,x,y,z}$ = 1 als plek x,y,z tijdens plaatsen van container i leeg is

Doelfunctie:

Df = min($\left(\sum_{i=1}^{I}\left(travel_{i,x} * C_{i,x,y,z}\right)\right)$)

Restricties:

1. $C_{i,x,y,z} \leq 1 - E_{i,x,y,z-1}$
2. $C_{i,x,y,z} \leq E_{i,x,y,z}$
3. $C_{i,x,y,z} \leq \frac{1}{Z}\sum_{l=1}^{Z}(1 - E_{i,x,y-1,l})$
4. $LD_{i,v} * C_{i,x,y,z} \leq \frac{1}{Z-(z-1)}\sum_{u=z}^{Z}(E_{i,x,y-1,u} + \sum_{w=v}^{V}\sum_{l=1}^{i-1}(LD_{l,w} * C_{l,,x,y-1,u}))$
5. $\sum_x \sum_y \sum_z C_{i,x,yz} = 1$
6. $E_{i,x,y,z} = C_{i,x,y,z} + E_{i+1,x,y,z}$
7. $LD_{i,v} * C_{i,x,y,z} \leq \sum_{l=1}^{i-1}\sum_{w=v}^{V}(LD_{l,w} * C_{l,,x,y,z-1})$

Definitie restricties:

1. Geen zwevende containers (er moet een container onder de geplaatste container staan).
2. Container moet op een lege positie komen
3. Containers worden pas geplaatst op een y locatie als de y locatie erachter tot de maximale hoogte gevuld is. (Stel container i wordt geplaatst op (x,y) locatie (1,2), mag dat pas als (1,1) tot de maximale hoogte gevuld is)
4. Containers mogen alleen op plekken gezet worden als de containers erachter voor schepen bestemd zijn die op een later schip weggaan
5. Alle containers moeten op een plek geplaatst worden.
6. Als een container op plek x,y,z geplaatst wordt is deze daarna niet meer leeg
7. De containers die onder de container staan moeten voor latere schepen bestemd zijn. (Als er op een bepaalde (x,y,z) locatie een container voor schip 4 staat. Mag één z hoger de container maximaal voor schip 1,2,3, én 4 bestemd zijn

De doelfunctie plaatst de containers de containers dan in rijtjes over de y-as waarbij de X-coördinaat zo laag mogelijk is.

Restricties 1,2, 5 en 6 zijn hard gecodeerd in het RL-model. Restrictie 3 komt leert het vanzelf, omdat de kade daar altijd maximaal gevuld wordt. De 4e en 7e restrictie zijn puur voor het LP-model, aangezien het RL model niet weet welke container wanneer moet vertrekken.

Dit model is zowel in de Docplex (gratis versie) en de Guirobipy (academische versie) solvers van python opgesteld en door de bijbehorende solvers opgelost.

4 Resultaten

Paramaters

Om te kijken welke parameters het beste werken voor het RL-model is er gekeken naar een range van de volgende parameters:

- Learning rate: [0.001, 0.003, 0.01, 0.03, 0. 1]
- Batch size: [16, 24, 32, 64, 128]

Tijdens eerdere tests is gebleken dat het mogelijk is om het model goed te trainen in ongeveer 1000 games. Er is gekozen om voor elke combinatie van de bovenste waardes een run van 1000 games te doen, omdat na 1000 games voor alle combinaties duidelijk is hoe goed het model traint. Alle resultaten van de trainingen zijn in een grafiek gezet om te kijken welke combinaties van learning rate en batch size voor de hoogste gemiddelde score en dus de beste resultaten geven. Uit de resultaten is gebleken dat een learning rate van 0.001 ervoor zorgt dat het model sneller een hogere score behaalt. Ook is te zien dat een hogere batch size een hogere gemiddelde score bereikt, maar een hogere batch size zorgt er ook voor dat de trainingstijd groter wordt. De hoogst haalbare score was ongeveer 450. De combinatie van een learning rate van 0.001 en een batch size van 128 behaalde deze score het snelste en eindigde met de hoogste gemiddelde score (68.82). Bij een learning rate die hoger is dan 0.001 werd gelijk duidelijk dat het model niet goed traint en de lijn direct naar beneden gaat.

Opstelling

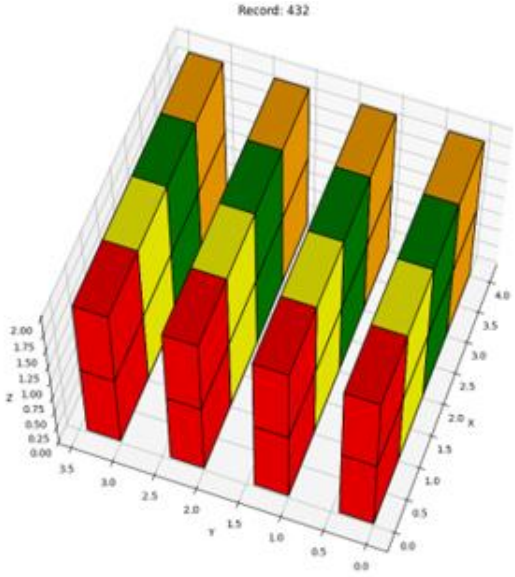
Een goede opstelling wordt binnen ons model gezien als een game waarbij de som van de scores per zet het hoogste is. De belangrijkste punten voor een goede score zijn:

- Containers voor hetzelfde schip in dezelfde rij
- Containers voor hetzelfde schip bovenop elkaar
- Zo min mogelijk containers voor andere schepen afdekken
- Zo min mogelijk vakken onbereikbaar maken voor de trucks

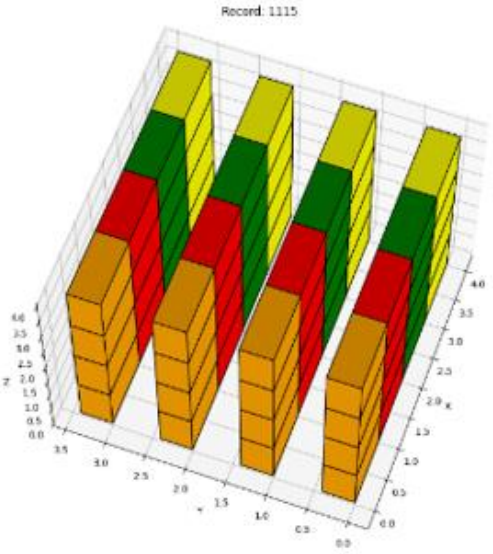
Eerst is er gewerkt met een vrij eenvoudige situatie. Er is gekeken naar een grid van 4x4x2 (zie figuur 2) waarbij alle rijen precies gevuld kunnen worden. Het lukt het model binnen 30 seconden om tot de opstelling in de eerste afbeelding te komen. Alle vakken zijn gevuld en de containers voor alle schepen zijn bereikbaar zonder containers van andere schepen te moeten verplaatsen.

Vervolgens is er gekeken naar een 4x4x4 (zie figuur 3) waarbij ook alle rijen volledig gevuld kunnen worden. Ook hier was te zien dat het model erg snel tot de perfecte opstelling komt, dit is te zien in de tweede afbeelding. Het lukte het model om tot de perfecte oplossing te komen binnen 45 seconden. De hoogte ten opzichte van het de eerste afbeelding is dus 2x zo groot, maar de trainingstijd is maar 1.5x zo groot.

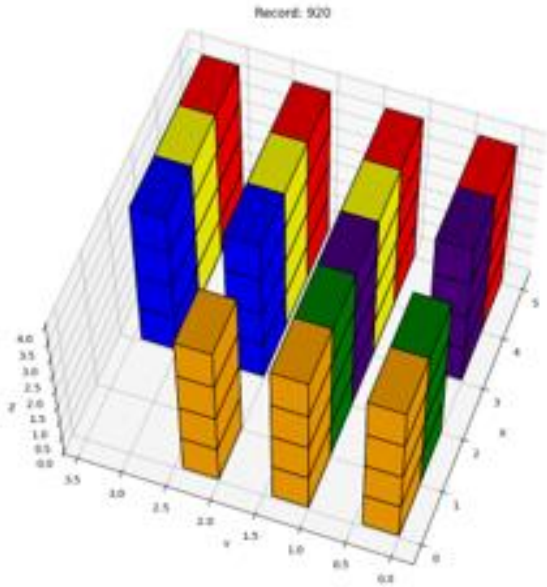
Om te kijken hoe het model omgaat met een reële complexere situatie is er ook gekeken naar een grid van 5x4x4 (zie figuur 4) waarbij de rijen niet perfect gevuld kunnen worden. Het model moet dus zelf uitzoeken wat de optimale opstelling is. Na ongeveer 2 minuten komt het model tot opstelling in de derde afbeelding. Hier is ook te zien dat de containers voor alle schepen bereikbaar zijn zonder containers van andere schepen hiervoor te hoeven verplaatsen. De containers van dezelfde schepen zijn ook netjes op elkaar gestapeld en er zijn geen lege vakken afgesloten. Dit kan dus gezien worden als een perfecte oplossing volgens de rewards en penalties die binnen het environment zijn opgesteld.



Figuur 2: 4x4x2



Figuur 3: 4x4x4

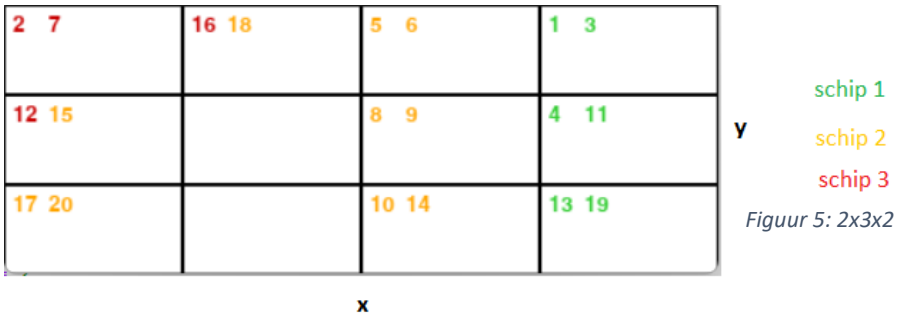


Figuur 4: 5x4x4

Docplex

In de komende afbeeldingen geeft het nummer van de container aan wanneer deze in de tijdsvolgorde van het schip gehaald wordt. De kleur geeft aan wanneer deze op een nieuw schip geladen wordt (eerst groen, dan oranje dan rood).

Voor de gratis licentie van docplex geldt een restrictie van maximaal 1000 variabelen en 1000 restricties. In figuur 5 zijn 20 containers geplaatst met een random vertrekschip voor elke container. De rechter twee rijen zijn gevuld volgens de beginstap. De linker 2 rijen zijn met behulp van het LP-model ingevuld. Dit zijn 8 containers, over een grid van 2x3x2, en 3 verschillende schepen om op te laden. Dit komt uit op 192 variabelen, en 728 restricties. Daardoor is dit bijna de maximale grootte die de solver aankan.

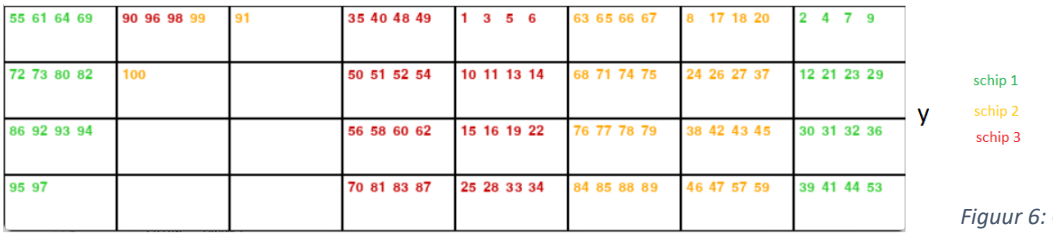


Figuur 5: 2x3x2

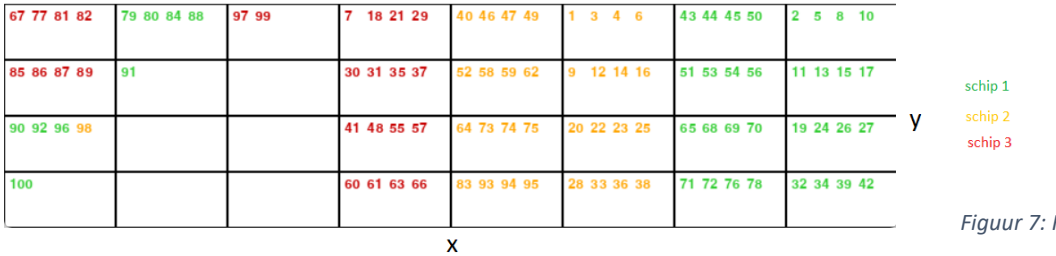
Gurobipy

De academische Gurobipy library kan genoeg variabelen en restricties aan om zowel het grid als het aantal containers te vergoten. Het model is hier ook in geprogrammeerd en levert de resultaten in onderstaande afbeeldingen. Het LP-model deelt hier 20 containers in over een grid van 3x4x4. Dit komt uit op 1920 variabelen en 15012 restricties. Het model doet er ongeveer 4 seconden over de oplossing hiervoor te vinden.

Zodra er een feasible oplossing is, verkrijgt het model ongeveer de helft van de tijd de optimale oplossing hiervoor. De andere helft van de tijd zitten er één of enkele opmerkelijke plaatsingen tussen die het model niet toe zou moeten staan, zoals in figuur 7 te zien is bij container 98. Dit blijkt uit 20 runs waarvan er in 9 gevallen een infeasible oplossing als optimaal uitkwam, terwijl er wel een feasible oplossing beschikbaar was.



Figuur 6: Correcte indeling van het veld



Figuur 7: Foute indeling van het veld

5 Discussie

Reinforcement Learning

Door gebruik te maken van een RL-model kan het positioneringsproces van de containers aangepakt worden, door het eerst kleinschalig te optimaliseren en dan op te schalen. Het environment van het huidige RL-model is maar een klein deel van de volledige kade. Uit de resultaten is gebleken dat een learning rate van 0.001 en een batch size van 128 het beste werken voor een kleinschalig environment, maar er is niet gekeken naar hoe deze waardes werken op de volledige kade. Voor het verwerken van de volledige kade in het model zal het environment dan ook uitgebreid moeten worden. De resultaten die behandeld zijn, zijn de resultaten van het model op een kleine schaal, hierdoor weten wij niet wat de haalbaarheid op grote schaal is.

Tijdens dit onderzoek is maar één model getest en dat is Deep Q Learning (DQN). Dit is uitgaande van het 'no free lunch theorem' niet vanzelfsprekend het beste model en de mogelijkheid bestaat dat een ander model betere resultaten boekt. Dit is tijdens ons onderzoek niet getest. Verder traint het model momenteel vanuit een leegstaande kade en wordt dan steeds aangevuld. Er zijn in dit onderzoek geen trainingen uitgevoerd op een gedeeltelijk gevulde kade. Voor het werken naar een realistische situatie zal het model uitgebreid moeten worden voor het trainen op een gedeeltelijk gevulde kade.

De resultaten tonen dat het RL-model met 1000 games relatief snel traint, waarbij zelfs op een 5x4x4 opstelling de ideale oplossing na ongeveer twee minuten wordt gevonden. Daarnaast worden toekomstige voorspellingen binnen een fractie van een seconde gedaan. Verder is het mogelijk het model op te schalen om de werkelijke situatie in de haven na te bootsen, maar dit zal ten koste gaan van de traintijd en het is onbekend in hoeverre het model nog bruikbaar blijft door de toenemende complexiteit.

Lineair programmeren

Door de manier waarop de volledige rijtjes ingevuld worden met containers voor hetzelfde schip wanneer dit mogelijk is, wordt het LP-model infeasable. Dit zou kunnen veranderen als er voor het indelen van de containers in de beginstap andere containers gekozen worden.

Daarnaast worden containers momenteel van rechts naar links gevuld, wat gedaan is om vanaf een bepaalde kant te kunnen plaatsen. Het ideale zou zijn om de containers zo dicht mogelijk bij de locatie neer te zetten waar het schip dat ze komt halen aanmeert. Dit zou zorgen voor minder rijtijd van de kranen tijdens het inladen. Dit is momenteel niet het geval.

Voor simplificatie en technische limiteringen kan er in dit model maar containers vanaf één zijde geplaatst worden in plaats van twee zijdes in de realiteit. De hoofdreden dat hiervoor gekozen is, is dat dit zorgt voor een simpeler model. Daarnaast zorgt dit ervoor dat je meer restricties overhoudt voor het plaatsen van containers. Dit zorgt er echter wel voor dat de indeling niet volledig gebruik maakt van de mogelijkheden in de realiteit. Een voordeel hiervan wel is dat de truck nooit om hoeft te rijden om de achterkant te bereiken.

Een volgend discussiepunt is dat er inconsistentie is in de rekentijd van het LP-model. Vaak kan het model de indeling uitrekenen binnen tien seconden, maar soms doet deze er meerdere uren over. Deze inconsistentie is natuurlijk niet handig en zorgt voor nadelen mocht dit in de realiteit gebruikt worden.

De duur van het uitvoeren van het LP-model gaat ook erg snel met een gemiddelde tijd van 4 seconden op een indeling van 3x4x4. Helaas is het mogelijk dat er in ongeveer 45% van de gevallen een onmogelijke oplossing wordt gegeven, wat het gebruik ervan minder aantrekkelijk maakt. Daarnaast is het schalen van het model naar grotere indelingen gelimiteerd aan de restricties van de Python Libraries die zijn gebruikt, wat het ingewikkeld maakt de werkelijke situaties van de haven te simuleren.

6 Conclusie

Uit de resultaten van het onderzoek naar de geschiktheid van Reinforcement learning en lineair programmeren voor het optimaliseren van het stapelingsproces in de haven is gebleken dat het RL-model consistenter is dan het huidige LP-model, echter kan er geen uitspraak worden gedaan over de bruikbaarheid in een reële situatie doordat de uitwerking op zeer kleine schaal is uitgevoerd.

7 Future work

Het onderzoek kan voortgezet worden door toevoegingen te doen aan zowel het LP- als het RL-model. Het op elkaar laten aansluiten van beide modellen kan als volgende stap wellicht leiden tot betere prestaties bij het trainen, door de oplossing van het LP-model als basis te gebruiken voor het RL-model.

Beide modellen kunnen uitgebreid worden door de resultaten toe te passen op het gebruik van een groter stuk van de kade in plaats van een enkel vlak, waarbij ook de hoeveelheid containers en schepen toenemen. Daarnaast kunnen de modellen worden uitgebreid door het te trainen op situaties waarbij een deel van de kade al gevuld is met containers, aangezien een kade nooit volledig leeg zal staan. Verder is het aan te raden de modellen te verifiëren met historische data van op de opdrachtgever.

Verder is het noodzakelijk ook het inlaadproces van de geplaatste containers naar andere schepen op te nemen, aangezien nu alleen het uitladen van de containers naar de kade is onderzocht, net als het inzetten van meerdere kranen voor het verplaatsen van containers. Ook onderzoek naar toepassing van andere RL- technieken is nodig, om te achterhalen of een bepaalde techniek niet beter werkt dan DQN.

Voor het LP-model is er onderzoek nodig om te achterhalen wat de ‘infeasible’ oplossingen veroorzaakt, waarbij gekeken kan worden of een bepaalde relaxatie binnen het model verantwoordelijk is. Daarnaast kan de plaatsing van de containers worden aangepast, zodat containers die spoedig weer opgeladen dienen te worden in eerste instantie dichterbij worden geplaatst.

8 Referenties en gerelateerde werk

Cofano. (sd). Opgehaald van Cofano.nl

GeeksForGeeks. (2022, 8 22). *What is Reinforcement Learning*. Opgehaald van GeeksForGeeks: https://www.geeksforgeeks.org/what-is-reinforcement-learning/

Kim, B. J. (2020, 4 20). *Spatial arrangement using deep reinforcement learning to minimise rearrangement in ship block stockyards.* Opgehaald van www.tandfonline: https://www.tandfonline.com/doi/full/10.1080/00207543.2020.1748247

Verma, R. S. (2019). *A Reinforcement Learning Framework for Container Selection and Ship Load Sequencing in Ports.* Delhi: TCS Research.

Wu, P. A. (2021, 6 9). *IS REINFORCEMENT LEARNING RIGHT FOR YOUR AI PROBLEM?* Opgehaald van MIT.edu: https://professional.mit.edu/news/articles/reinforcement-learning-right-your-ai-problem

Kanade, V. (2022, September 29). What Is Reinforcement Learning? Working, Algorithms, and Uses. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-reinforcement-learning/

Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. Transportation Research Part B: Methodological, 37(10), 883–903. https://doi.org/10.1016/s0191-2615(02)00089-9

9 Begrippenlijst

Data science	Het halen van kennis uit data door inzet van statistieken, analyses en machine learning.
Machine learning	Computerprogramma dat leert een taak uit te voeren op basis van ervaringen uit data zonder expliciete instructie .
Reinforcement learning	Een machine learning programma dat met data zonder labels op basis van ervaringen en beloningen leert voorspellingen te doen.
Linear programmeren	Een methode om optimalisatieproblemen op te lossen.
Containeroverslag	Het verplaatsen van containers tussen schepen en de havenkade.
Generalisatie	Hoe goed het model werkt op nog onbekende data.
DQN	Een Reinforcement Learning algoritme dat de waarde van een bepaalde actie kan leren in een bepaalde state.
Feasible	Term die aanduidt dat er minstens één oplossing bestaat die aan alle restricties voldoet.
Gradient	Een afgeleide van een functie die de verandering in alle gewichten meet met betrekking tot de verandering in fout.