1. **Doelstelling**
   De eerste periode van dit project doel formuleren en literatuuronderzoek. We besloten ons te richten op het optimaal indelen van een yard in de terminal voor het zo snel mogelijk inladen op de zeevaartschepen, die containers op komen halen. We zouden dan geen rekening houden met de binnenvaartschepen die de containers naar de terminal brengen en de volgorde van binnenkomst. Met de groep hebben we een hoofdvraag geformuleerd, maar geen duidelijke deelvragen, dus deze heb ik zelf gekozen.

   De hoofdvraag: Hoe kan de indeling van yards op containerterminals geoptimaliseerd worden door middel van geautomatiseerde methodes, zodat zeevaartschepen zo snel mogelijk weer kunnen vertrekken?
   a. Welke data is er beschikbaar?
   b. Welke optimalisatie methodieken/modellen zijn er al ontwikkeld voor soortgelijke problemen?
   c. Welke factoren en restricties spelen een rol bij het optimaliseren van dit terminalproces?
   d. Welke modellen kunnen rekening houdend met de gevonden factoren gebruikt worden voor het automatiseren van de indelingsmethode?

   Daarnaast bedachte we gedurende het project dat het ook een belangrijk doel is om een schaalbaar model te maken. Het Container Stacking Problem (CSP) is namelijk een dynamisch probleem en in het echt heb je te maken met veel grotere yards en meer containers. We wilde klein beginnen (3x3) en dan kijken of we het model goed uit konden breiden naar meer containers en een grotere yard (van 2D naar 3D onder andere).

2. **Data Preprocessing (deelvraag a)**
   Er was veel data beschikbaar van Cofano, die wij bekeken hebben. Deze data was echter niet nuttig voor onze doelstelling en aanpak. We besloten zelf data te simuleren.

   De gesimuleerde data was een lijst waarin de containers stonden die in de yard geplaatst moesten worden en voor welk zeevaartschip de containers bestemd waren. Het nummer van de container stond voor het schip, die ze zou komen ophalen. De containers stonden geshuffeld in de lijst, omdat er in dit geval niks bekend is over de volgorde waarop containers binnen komen en we hier geen rekening mee houden in ons onderzoek. Wel hebben we aangenomen dat op een binnenvaartschip containers kunnen staan die voor verschillende zeevaartschepen bestemd konden zijn.

3. **Reinforcement Learning**
   Het literatuuronderzoek, wat ik gedaan heb naar verschillende methodes om het CSP aan te pakken, is te vinden in hoofdstuk 2.2 alinea 1 **(deelvraag b)**.
   Na allemaal onderzoek te hebben gedaan en gepraat te hebben met medestudenten en de docenten, zijn we ons gaan verdiepen in Reinforcement Learning. Door middel van literatuuronderzoek (zie hoofdstuk 2.2 alinea 2) ben ik gaan begrijpen hoe dit precies werkt.

   (deelvraag c:)
   Voordat we een model gingen kiezen en trainen, moesten we vaststellen welke factoren invloed hebben op het optimaal indelen en welke eisen/restricties daarbij komen kijken.
   Restricties:

- Niet een container plaatsen waar al container staat.
- Niet een container in de lucht plaatsen
- Niet een container buiten de yard plaatsen
- Containers kunnen alleen vanaf de lange zijde gepakt worden
Andere factoren van belang bij optimaal indelen
- Het is belangrijk dat containers niet ingeboxed staan (dat is als 2 containers voor ander schip aan beide lange zijdes van de container staan of als er een container op een container voor ander schip staat)
- Stacker heeft zo min mogelijk zetten nodig om bij container te komen

Vervolgens zijn we samen gaan brainstormen over een gepaste reward functie. Vervolgens gingen we allemaal zelf een reward functie maken los van elkaar en dan vergelijken. We speelden allemaal 1 game en berekende daarbij de reward. Onder 1 game verstonden we het invullen van een 3x3 yard, door 9 zetten uit te voeren. 1 zet is het plaatsen van 1 container in de yard.

Een ander groepsgenoot heeft onderzoek gedaan naar welk RL-model geschikt was voor dit probleem **(deelvraag d)** en dat bleek PPO te zijn. We hebben ook met Jesse gesproken, die ons ook PPO aanraadde.

De volgende stap was een RL-model maken voor een 3x3 grid. We gingen weer los van elkaar een environment maken en een model trainen. Eerst met alleen de restrictie dat er geen containers op dezelfde plek geplaatst werden en toen dat werkte gingen we onze uitgebreidere reward functie erin verwerken. Mijn reward functie bleek uiteindelijk te ingewikkeld en heb ik wat in moeten korten, voor een snellere training van het model.

Tijdens het trainen van onze modellen hebben we wat vertraging opgelopen. Dit kwam door de fout dat we eerst niet de volgende container meegaven aan de observation space. Het is natuurlijk wel belangrijk dat het model weet welke container die gaat plaatsen, zodra die een keuze maakt waar die de container gaat zetten. Anders is het moeilijk om te leren. Dit was een stomme fout, maar gelukkig ontdekte Jeroen dit en konden we verder.

Voorbeelden van wat ik gedaan heb tijdens het werken aan ons model:

a. Ik heb een groot deel van het Literatuuronderzoek gedaan. Iedereen ging zelf codes van RL modellen en videos bekijken, maar ik was verantwoordelijk voor het ‘nette’ literatuuronderzoek wat ook in het paper is gekomen. Wat ik gedaan heb om kennis op te doen staat omschreven in hoofstuk 2.2.
b. Ik heb in een notebook een reward functie gebouwd door een game te spelen en per zet een reward of penalty mee te geven. In dit notebook kan je precies zien hoe stap voor stap een yard van 3x3 wordt ingevuld en waar penalty’s en rewards voor worden gegeven. Dit heb ik gebruikt om te oefenen met het maken van een goede rewardfunctie. Ik kon zo bijvoorbeeld zien dat hoge penalty’s of rewards geven soms niet representatieve eindscores gaf en heb besloten in mijn RL-model niet te werken met hele hoge waardes in de reward functie. Hoe ik dit heb gedaan is te zien in notebook ‘Reward functie 3x3 yard- 1 game’ in hoofdstuk 3.2.
c. Vevrolgens heb ik een simpele environment van 3x3 containeryard ontworpen en een RL model gerund, waarbij het alleen trainde op het niet dubbel plaatsen van containers. Dit lukte erg snel.

De observation space is een box van 3 bij 3 (volgende container meegeven is nog niet belangrijk, omdat het hier nog niet gaat om optimale plaatsing vinden), de action space zijn de coordinaten x en y van waar een container geplaatst gaat worden en er is een lijst met containers die geplaatst moeten worden. Het model kan na 100000 timesteps (en misschien wel sneller, dat heb ik niet geprobeerd) alle 9 containers op een eigen plek plaatsen.

d. Daarna heb ik dit uitgebreid naar het optimaal indelen van een 3x3.
e. Ik ging als eerste van mijn groep de hoogte in uitproberen. Ik begon simpel met een 2x2x2. De reward moest ik aanpassen en in de observation space werd de box veranderd in een numpy array.

Een andere groepsgenoot heeft daarna succesvol een 3x3x3 ingedeeld vanuit leeg en met dat model zijn we verder gegaan voor ons paper

*Hieronder heb ik een klein stukje resultaten, conclusie en discussie/aanbevelingen over **mijn eigen** werk geschreven. Als groep hebben we meer bereikt, zoals terug te lezen is in het paper (zie hoofdstuk …).*

4. **Resultaten**
    a. Een 3x3 indelen met 9 containers van 3 verschillende zeevaartschepen (3 per schip), zoals terug te zien in het notebook '3x3 yard model' in hoofdstuk 3.2. Het model is getrained in 155 secondes
    b. Een 2x2x2 ingedeeld met 8 containers van 4 verschillende zeevaartschepen (2 per schip), zoals terug te zien in het notebook '2x2x2 yard model' in hoofdstuk 3.2. Dit model is getrained in 646 secondes.

5. **Conclusie**
    a. Hier is een optimale indeling voor een 3x3 yard. De stacker kan nu namelijk overal bij, onafhankelijk van welk zeevaartschip eerst binnen komt.
    b. Hier is een optimale indeling voor een 2x2x2 yard. De stacker kan nu namelijk overal bij, onafhankelijk van welk zeevaartschip eerst binnen komt.
    Hiervoor moest de reward functie aangepast worden, omdat er andere restricties bij kwamen kijken. Ook veranderde de yard in de observation space van een box naar een numpy array. En kwam er in de action space een z-coordinaat bij.

    Om terug te komen op de hoofdvraag: De indeling van een yard op een container terminal kan optimaal ingedeeld worden door een Reinforcement Learning model, het PPO-model, zo dat de zeevaartschepen zo kort mogelijk aan de kade liggen. De stackers kunnen overal in 1 keer bij, onafhankelijk van welk schip eerst aan komt.

    Over de schaalbaarheid: Het model is succesvol uitgebreid van een 3x3 naar een 2x2x2, maar wel met de nodige aanpassingen in de reward functie en de observation en action space.

6. **Discussie**
    a. Het model heeft best veel aanpassingen nodig als er iets veranderd in de situatie. De containeryard uitbreiden of een andere containerlijst mee geven, zal niet gaan zonder handmatig het model en de reward aan te passen. Dit komt omdat de reward functie die ik nu heb geschreven heel specifiek zich richt op de huidige situatie en de

environment anders is voor 2D en 3D bijvoorbeeld.
Aanbeveling: In het model staan dingen vast, zoals de lijst met containers en de grootte van de yard. Als je het probleem uit breidt, moet je dit steeds handmatig aanpassen. Voor in het vervolg is het handig als dingen , zoals breedte, lengte, hoogte van de yard, het aantal schepen en aantal containers per schip, variabel zijn. Zo hoeven ze alleen aan het begin aan het model meegegeven te worden en is het model makkelijker uit te breiden.
Aanbeveling: De reward functie zou zo geschreven moeten worden dat deze bruikbaar is voor verschillende groottes van yard en hoeveelheden containers en schepen. Een generieke reward functie die penalty's en rewards geeft die niet afhankelijk zijn van de mate en dimentie van de yard, zal beter werken bij complexere situaties. Bijvoorbeeld door te kijken naar inboxing en het plaatsen van dezelfde containers bij elkaar. Ook is het een idee om de zetten die een stacker nodig heeft mee te nemen in het model.

b. Het model traint nog niet heel snel (de 2x2x2 doet er ruim 10 minuten over). Er is nog geen onderzoek gedaan om dit te verbeteren en geen learning curve geplot, omdat het voor deze kleine situaties geen problemen opleverde. Als het probleem complexer wordt, wordt dit wel onhandig.
   Aanbeveling: Als eerst is het aan te raden een learning curve te plotten waarin te zien is hoe snel het model leert en of daar opmerkelijke dingen in terug te zien zijn. Ook kan het zijn dat het model al lang zijn optimum bereikt heeft en onnodig door traint. Door hier onderzoek naar te doen, is het misschien mogelijk het model voor minder timesteps te laten runnen.
   Aanbeveling: Daarnaast kan het helpen om het model ook te trainen vanaf een deels ingevulde yard. Zo kan je het model eerst trainen op een 90% gevulde yard, dan een 80% gevulde enzovoort. Als het model moeite heeft met het vinden naar een goed oplossing, kan dit het misschien wat helpen.
c. Aanbeveling: Op dit moment heb ik zelf niet geëxperimenteerd met verschillende RL-modellen. In vervolgonderzoek is het interessant om dit op te pakken. Mogelijk is er een model wat beter presteert voor dit probleem.