Afstudeerscriptie

Sjors Jansen
12103217

Logistiek probleem
Voorsorteren voor Anthura Arndt

Sjors Jansen

Maasdijk

26-8-2016

Logiqs B.V.

Voorwoord

Als afsluiting van mijn vierde jaar van de opleiding Toegepaste Wiskunde is er een afstudeerstage gelopen. Tijdens deze stage is er een opdracht uitgevoerd en over deze opdracht heb ik een scriptie geschreven.

Een klant van Logiqs B.V. inspireerde het bedrijf tot het uitgeven van deze opdracht. Deze klant had een logistiek probleem waar Logiqs B.V. graag een oplossing voor wilde bedenken.

In deze scriptie is uitgebreid beschreven wat dit probleem was, wat de stappen waren tot het oplossen van het probleem en wat de uiteindelijke oplossing is.

De scriptie is bedoeld voor mijn begeleiders tijdens mijn afstuderen. Zowel meneer Van Noort, mijn afstudeerbegeleider van de Haagse Hogeschool, als Koen van Leeuwen, mijn begeleider bij Logiqs B.V., krijgen het te lezen. Daarnaast komt de scriptie online te staan en kunnen studenten deze gebruiken als inspiratiebron.

Als laatste wil ik graag een aantal mensen bedanken die geholpen hebben bij het werken aan de opdracht. Allereerst wil ik Gert-Jan van Staalduinen, directeur van Logiqs B.V., bedanken voor het verschaffen van een afstudeerplek. Hij gaf mij alle ruimte om te doen waarvoor ik gekomen ben. Daarnaast wil ik mijn bedrijfsbegeleider, Koen van Leeuwen, bedanken voor de begeleiding vanuit het afstudeerbedrijf. Als laatste wil ik meneer Van Noort bedanken voor het begeleiden vanuit de Haagse Hogeschool.

De Lier, augustus 2016

Inhoudsopgave

Samenvatting

Achtergrondinformatie:

Logiqs B.V. produceert al jaren logistieke systemen voor de glastuinbouw. Deze systemen verplaatsen de planten automatisch door de kas heen. De planten bevinden zich op een rolcontainer. Een rolcontainer is een metalen rechthoekige constructie met opstaande rand. Een rolcontainer verplaatst zich over twee buizen. De twee buizen samen worden een pijpenbaan genoemd. Door het geven van een opdracht aan Dat-A-Logiqs, een softwareprogramma van Logiqs B.V., worden de rolcontainers verplaatst.

Het verplaatsen van een rolcontainer wordt gedaan door de transportbanen en de shuttles. De transportbanen bevindt zich aan de uiteinden van de pijpenbanen en verplaatsen de rolcontainers in de breedte door de kas. De shuttles rijden onder de pijpenbaan door en verplaatsen de rolcontainer zo in de lengte van de kas over de pijpenbanen heen.

Probleemstelling:

Anthura Arndt is één van de klanten van Logiqs B.V. Zij hebben een probleem in de kas wanneer zij deze vol zetten met rolcontainers. Elke dag moeten er voor de verkoop planten naar een specifieke locatie in de kas verplaatst worden. Dit heet voorsorteren. Omdat de planten willekeurig over de kas verspreid zijn, zijn er veel verplaatsingen nodig om de planten op de juist plaats te krijgen. **Het handmatig invoeren van al deze verplaatsingen duurt voor de tuinder vaak drie kwartier.** Daarnaast zorgt een volle kas ervoor dat er regelmatig rolcontainers dubbel verplaatst moeten worden. **Dubbel verplaatsen is inefficiënt.**

Afstudeeropdracht:

De afstudeeropdracht is het automatiseren van het voorsorteren. Dit moet gebeuren met behulp van een algoritme. Het algoritme moet het handmatig invoeren van de opdrachten overbodig maken. Daarnaast moet het algoritme het voorsorteren zo efficiënt mogelijk, zonder dubbele verplaatsingen, doen. Er is echter geen eis opgesteld over hoe efficiënt he algoritme moet zijn. **Wel is de volgende eis opgesteld: het algoritme moet voor iedere startsituatie een oplossing genereren.**

Analyse:

Geanalyseerd is hoe de kas van Anthura Arndt eruit ziet. Hieruit is gebleken dat de kas bestaat uit veertien pijpenbanen met op iedere pijpenbaan ruimte voor 42 rolcontainers. Er zijn echter maar veertig rolcontainers aanwezig per pijpenbaan. Dit betekent dat er 28 lege plaatsen zijn die gebruikt kunnen worden om dubbele verplaatsingen te voorkomen. Ook is uit de analyse bepaald dat de graadmeter voor de efficiëntie gelijk is aan het totaal aantal rolcontainerverplaatsingen.

Literatuuronderzoek:

In de literatuur is op zoek gegaan naar methoden die het probleem van Anthura Arndt op kunnen lossen. Uit de literatuur is gebleken dat er meerdere methoden zijn die dit kunnen. Alle methoden gebruiken een zoekboom bij het oplossen van het probleem. Uit de verschillende zoekmethoden is de Best-first search gekozen als beste zoekmethode. De Best-first search zoekt gericht naar de situatie met de beste uitgangspositie. Deze uitgangssituatie is gelijk aan een cijfer dat wordt bepaald door een combinatie van verschillende factoren. Verbeteringen op deze methoden zijn het beperken van opties en het verwijderen van dubbele situaties.

Beginalgoritme:

De Best-first search is de methode die het gerichtst zoekt naar de oplossing. Deze methode vindt als eerste een oplossing en daarom is er gekozen om deze te gebruiken in het algoritme. De factoren voor het bepalen van de uitgangspositie zijn bepaald door middel van veel testen met verschillende waarden. Omdat het probleem erg groot is kan alleen de Best-first search het probleem niet oplossen. Daarom is ook om het beperken van opties en het verwijderen van dubbele situaties in het algoritme in gevoerd.

Na het programmeren van het algoritme is deze op 57 verschillende startsituaties getest. Hierbij is getest op het algoritme voor alle tests een oplossing genereerde. Ook de gemiddelde tijdsduur en efficiëntie is berekend. De efficiëntie is bepaald door het verschil in het aantal verplaatsingen tussen de gevonden oplossing en de optimum oplossing nemen. Deze is uitgedrukt in een percentage en wordt de efficiëntieafwijking genoemd. Uit de test is gebleken dat het algoritme voor drie startsituaties niet tot een oplossing kwam. Dit betekent dat het algoritme nog niet optimaal functioneert. De gemiddelde tijdsduur van de tests was 73,86 seconden en de gemiddelde efficiëntieafwijking was 13,55%.

Na het testen is onderzocht waarom de drie startsituaties niet op een oplossing uitkwamen. Hieruit is gebleken dat de combinatie van factoren niet optimaal is. Na het aanpassen van de combinatie van factoren is er een tweede test gedaan. Bij deze tests genereerde het algoritme voor alle startsituaties een oplossing. Daarnaast is de gemiddelde tijdsduur gedaald naar 59,52 seconden dit is een versnelling van bijna vijftien seconden. Ook de gemiddelde efficiëntieafwijking is met bijna drie procent naar 10,59% flink gedaald. Dit betekent dat het verschil tussen de gevonden oplossing en de optimale oplossing verkleint is. Het algoritme genereert niet alleen voor alle startsituaties een oplossing maar doet dit ook nog sneller en efficiënter.

Wensen van Logiqs B.V.:

Omdat vanuit alle startsituaties een oplossing gegenereerd is, is aan de enige eis van Logiqs B.V. voldaan. Het doel van de afstudeeropdracht is hiermee bereikt. Samen met Gert-Jan van Staalduinen en Koen van Leeuwen zijn er voor het vervolg van de afstudeerperiode vijf wensen opgesteld. De eerste wens betrof het verbeteren van de efficiëntie van de oplossingen. De gemiddelde efficiëntieafwijking van de oplossingen mocht hoogstens tien procent zijn. Door het invoeren van meerdere strategieën in het algoritme is dit gerealiseerd. Een strategie is gedefinieerd als de combinatie van factoren die het cijfer van de uitgangspositie bepalen. Door deze factoren te veranderen ontstaan nieuwe strategieën. Als test zijn meerdere strategieën achter elkaar uitgevoerd. Uit de tests is gebleken dat de gemiddelde efficiëntieafwijking na de verandering gedaald is 7,72%. Hiermee is aan de wens eerste wens van Logiqs B.V. voldaan.

Als tweede wens wil Logiqs B.V. het algoritme in hun software implementeren. Koen van Leeuwen en Maarten van der Veen hebben de input en output van het algoritme gekoppeld aan de software. Deze koppeling zorgt ervoor dat het algoritme via de software van Logiqs getest kan worden.

De derde wens van Logiqs is het veranderen van de graadmeter voor de efficiëntie. Deze was eerder vastgesteld op het aantal rolcontainerverplaatsingen. Bij deze wens wordt deze vastgesteld op het aantal shuttleverplaatsingen. Een shuttleverplaatsing is een verplaatsing waarbij de shuttle van pijpenbaan veranderd. Door het invoeren van deze wens is het niet meer mogelijk de efficiëntieafwijking van de oplossingen uit te rekenen. Hierdoor is een goede vergelijking met de eerdere algoritmen niet meer mogelijk.

Na het doorvoeren van de derde wens is er geen tijd meer om ook de vierde en de vijfde wens in het algoritme in te voeren. Beide worden daarom als aanbeveling voor Logiqs B.V. gegeven. De wensen waren het voorsorteren voor meerdere dagen en het voorsorteren voor meerdere dagdelen.

Conclusie:

Geconcludeerd kan worden dat het gelukt is om een algoritme te maken dat het probleem van Anthura Arndt oplost. Het algoritme genereert voor iedere startsituatie een oplossing waarmee het aan de eis van Logiqs B.V. voldaan heeft. **Hiermee kan geconcludeerd worden dat het handmatig invoeren van de opdrachten overbodig geworden is.** Daarnaast is het gelukt om drie van de vijf wensen van Logiqs B.V. door te voeren in het algoritme. Hierna is de een efficiëntie van 13,55% met bijna zes procent gedaald naar 7,72%. Dit betekent dat het algoritme efficiënte oplossingen genereert.

Aanbevelingen:

Logiqs B.V. is aanbevolen om de laatste twee wensen alsnog door te voeren. De eerste wens is het voorsorteren naar meerdere dagen. De twee wens is het voorsorteren naar meerdere dagdelen. Daarnaast wordt Logiqs B.V. aanbevolen om het algoritme universeel te maken zodat het niet allen bij Anthura Arndt maar ook op andere klanten van toepassing is. Nu kan het algoritme alleen kassen aan met dezelfde indeling als bij Anthura Arndt. De afmetingen van de kas kunnen hierin wel verschillen. De laatste aanbeveling voor Logiqs B.V. is het aanpassen van het algoritme zodat het ook toepasbaar is op het palletopslagsysteem van Logiqs B.V.

Verklarende woordenlijst

Begrip	**Definitie/verklaring**
Algoritme	Een vaste reeks instructies die vanuit een beginsituatie altijd de eindsituatie bereikt.
Anthura Arndt	Het tuinbouwbedrijf dat een voorsorteerprobleem heeft en Logiqs B.V. naar een oplossing heeft gevraagd.
Automatische transportbaan	Een vaststaande machine die de rolcontainers in de breedte van de kas kan verplaatsen.
Beginsituatie	Dit is de beginstand van het probleem.
Beslissingsprobleem	Een probleem dat opgelost wordt na het maken van meerdere beslissingen achter elkaar.
Best-first search	Een zoekstrategie waarbij de volgende situatie bepaald wordt aan de hand van een cijfer.
Branch and Bound	Een zoekstrategie waarbij per situatie de toekomst voorspelt wordt in het ideale geval.
Breadth-first traversal	Een zoekstrategie waarbij per laag van de zoekboom gezocht wordt naar een oplossing.
Brute-force search	Een zoekstrategie waarbij alle mogelijke situaties doorzocht worden.
Dat-A-Control	Het volledige softwarepakket van Logiqs B.V.
Dat-A-Logiqs	Het verplichte onderdeel van het softwarepakket. Deze geeft het kasoverzicht weer.
Depth-first traversal	Een zoekstrategie waarbij per tak van de zoekboom naar een oplossing gezocht wordt.
Dictionary	Een object dat situaties kan opslaan met een sleutel. Deze sleutel is uniek.
Eindsituatie	Dit is de eindstand van het probleem.
Efficiëntieafwijking	Het verschil in verplaatsingen tussen de gevonden oplossingen en de optimale oplossing uitgedrukt in een percentage
First in, First out principe	Wat als eerst in de rij gaat komt er als eerste weer uit.
Foundations	Eindbestemming van de kaarten bij FreeCell.
FreeCell	Kaartspel van Windows.
FreeCells	De lege cellen in het spel FreeCell.
Greedy search	Een zoekstrategie waarbij gekeken wordt naar de beste situatie in de tak.
Groene rolcontainer	Een rolcontainer die niet voorgesorteerd hoeft te worden.
ICUBE	Het palletopslagsysteem van Logiqs B.V.
Input	De informatie die het algoritme nodig heeft om te kunnen draaien.

Intern transportsysteem	Alle machines in een kas te samen is het interne transportsysteem van de kas.
Laag	Verzameling van alle situaties in de zoekboom met hetzelfde aantal uitgevoerde verplaatsingen
Lijst	Een object waarin situaties opgeslagen kunnen worden in de volgorde die de gebruiker wil.
NP-volledige klasse	De klasse met problemen waarvoor geldt dat er geen algoritme is dat de optimale oplossing met zekerheid vindt.
Oplossing	De serie verplaatsingen die nodig is om van de beginsituatie in de eindsituatie te komen.
Output	De informatie die het programma geeft na uitvoering van het algoritme.
Pijpenbaan	Twee metalen buizen waar de rolcontainers zich op bevinden.
Queue	Een object waarin situaties opgeslagen kunnen worden volgens het first in, first out principe.
Rode rolcontainer	Een rolcontainer die voorgesorteerd moet worden.
Rolcontainer	Metalen rechthoekige bak met opstaande rand. Wordt gebruikt om planten op te telen.
Schuifpuzzel	Spel waarbij tegeltjes verschoven moeten worden tot deze op de juiste plek staan.
Shuttle	De machine die onder de pijpenbaan rijdt en rolcontainers op kan tillen en verplaatsen.
Shuttle-instructie (IN)	Instructie waarbij een rolcontainer door een shuttle de pijpenbaan in gebracht wordt.
Shuttle-instructie (UIT)	Instructie waarbij een rolcontainer door een shuttle de pijpenbaan uit gehaald wordt.
Shuttleverplaatsing	De verplaatsing van een shuttle naar een andere pijpenbaan of pijpenbaanuiteinde.
Situatie	Iedere mogelijke kasindeling in het proces.
Sourch Tree	Een online database waarin het hele bedrijf bestanden opslaat.
Tegeltje	Vierkantje met getal of plaatje dat verschoven moet worden om de schuifpuzzel uit te spelen.
Transportbaaninstructie	Instructie waarbij een rolcontainer over de transportbaan verplaatst wordt.
Visible	Een oplossing is visible als de verplaatsingen in de oplossing ook echt resulteren in het eindresultaat.
Visual Studio	Microsoft ontwikkeltools.
Voorsorteerbaan	Hier moeten de voor te sorteren rolcontainers naartoe verplaatst worden.
Zoekboom	De zoekboom geeft per verplaatsing weer wat de situaties zijn die hieruit voortkomen.
Zoekstrategie	Een methode van zoeken die de route door de zoekboom bepaald.

1. Inleiding

Al jaren maakt Logiqs B.V. logistieke systemen voor in de glastuinbouw. Deze systemen verplaatsen groepen met planten van de ene plek in de kas automatisch naar een andere plek in de kas. Voor iedere plantengroep die verplaatst wordt geeft de tuinder een opdracht aan het systeem waarna deze de opdracht uitvoert.

Een van de klanten van Logiqs B.V. is Anthura Arndt. Anthura heeft een probleem geconstateerd. Het probleem ontstaat wanneer zij een bijna volle kas hebben en meerdere plantengroepen wil verplaatsen die zich op verschillende andere plaatsen in de kas bevinden. Door te weinig ruimte in de kas moet de ene plantengroep dan wachten met verplaatsen, tot de andere plantengroep verplaatst is. Dit is inefficiënt en zorgt ervoor dat het verplaatsen langer duurt dan nodig.

Deze scriptie beschrijft een algoritme dat uitrekent welke opdrachten het systeem moet krijgen zodat er op een efficiënte manier groepen met planten verplaatst worden. Het algoritme heeft als output een efficiënte serie opdrachten.

Als eerste wordt in hoofdstuk 2 achtergrondinformatie gegeven over Logiqs B.V. Hierna wordt in hoofdstuk 3 een uitgebreide opdrachtbeschrijving gegeven. Hoofdstuk 4 bevat de uitwerking van de analyse op het logistieke systeem. In hoofdstuk 5 volgt het literatuuronderzoek. Hierna wordt het beginalgoritme in hoofdstuk 6 beschreven. Na het beginalgoritme wordt in hoofdstuk 7 het eindalgoritme gepresenteerd. Hierna is de conclusie getrokken en zijn er een aantal aanbevelingen gedaan. De scriptie eindigt met een literatuurlijst en de bijlagen.

2. Achtergrondinformatie

In dit hoofdstuk is beschreven waar Logiqs B.V. voor staat en wat de geschiedenis van het bedrijf is. Daarnaast behandeld het hoofdstuk meerdere producten van Logiqs B.V.

2.1 Waar staat Logiqs B.V. voor?

De naam Logiqs staat voor Logistics Quality Systems. Het bedrijf is gevestigd in Maasdijk en is al ruim 35 jaar producent van logistieke systemen in de glastuinbouw. Logiqs B.V. ontwerpt, produceert en installeert interne transportsystemen voor kwekerijen van planten, bloemen en groenten. Ze maken het voor tuinders mogelijk om met zo weinig mogelijk personeel een zo groot mogelijk bedrijf te runnen. Met de slogan: “We think logistics” wil Logiqs B.V. een nieuwe dimensie geven aan de glastuinbouw.

2.2 De geschiedenis van Logiqs B.V.

De naam Logiqs B.V. is niet de eerste naam van het bedrijf. Bij de oprichting heette het bedrijf Alcoa Agro/In transit. Alcoa Agro/In transit was toen gevestigd aan de Lierweg in De Lier. Na het faillissement in 2009 is er een doorstart gemaakt. Het bedrijf verhuisde naar het Honderdland 131 in Maasdijk en veranderde de bedrijfsnaam in Logiqs Agro International. Vijf jaar later ging Logiqs Agro International samen met haar zusterbedrijf Leen Huisman voor een tweede keer failliet. Dit keer scheidde Logiqs Agro zich af van Leen Huisman en maakt opnieuw een doorstart. Dit maal onder de naam Logiqs B.V. Dit is de huidige bedrijfsnaam. Sinds de tweede doorstart van het bedrijf gaat het erg goed. Het bedrijf is in een jaar tijd van dertien werknemers gestegen naar meer dan dertig werknemers. Dit aantal groeit nog steeds.

2.3 De producten van Logiqs B.V.

Logiqs B.V. ontwerpt en produceert meerdere producten die de logistiek in de glastuinbouw verbeteren. Van vier van deze producten is het belangrijk te weten wat ze inhouden omdat ze regelmatig in de scriptie voorkomen. Daarnaast is het belangrijk om kennis te nemen van de software die Logiqs B.V. Daarom zijn de vier producten en een deel van de software in dit hoofdstuk uitgeschreven.

2.3.1 Rolcontainer

De rolcontainer is het product dat Logiqs B.V. het vaakst produceert. Een rolcontainer is een metalen constructie waarop de teelt van de planten plaatsvindt. De rolcontainers bevinden zich in de kas op twee lange buizen. Twee van deze buizen heten samen een pijpenbaan. Het verplaatsen van de rolcontainers gebeurt over deze pijpenbanen.

Omdat iedere kas andere afmetingen heeft en iedere kas daarom een andere indeling en grootte heeft, berekent Logiqs B.V. voor iedere kas het ideale formaat voor de rolcontainers Hierdoor gebruikt de tuinder het kasoppervlak optimaal. In figuur 2.3.1.1 is een afbeelding van een rolcontainer te zien.

Figuur 2.3.1.1: Een rolcontainer met gaasbodem

Naast het makkelijk verplaatsen van grote partijen planten biedt de rolcontainer de tuinder de mogelijkheid van meer teeltlagen. Meer teeltlagen resulteert in een optimale ruimtegebruik.

2.3.2 Shuttle

Omdat de pijpenbanen en de rolcontainers geen motoren hebben kunnen deze niet automatisch bewegen. Logiqs B.V. heeft daarom een machine ontwikkeld die de rolcontainers over de pijpenbanen heen verplaatst. Deze machine noemt Logiqs B.V. een shuttle.

De shuttle rijdt onder de pijpenbaan door en verplaatst de voorste rolcontainer van de pijpenbaan naar het uiteinde van de pijpenbaan. Hier neemt een automatische transportbaan, beschreven in paragraaf 2.3.3, de rolcontainer over van de shuttle.

De shuttle behoord niet toe aan één pijpenbaan, maar kan zich over meerdere pijpenbanen bewegen. De shuttle kan zich hierdoor in de lengte en de breedte van de kas bewegen. Het verplaatsen van rolcontainers kan de shuttle echter alleen in de lengte over de pijpenbanen heen. Het in de breedte verplaatsen van een rolcontainer wordt door de transportbaan gedaan.

Het in de breedte verplaatsen van een shuttle gebeurt wanneer de shuttle naar een andere pijpenbaan moet. Doordat de shuttle over meerdere pijpenbanen kan bewegen, bewerkt een shuttle een groot kasoppervlak.

Er zijn drie soorten shuttles. De eerste shuttles kan één rolcontainer tegelijk verplaatsen. De tweede kan twee rolcontainers verplaatsen en de derde soort kan drie rolcontainers tegelijk verplaatsen. In figuur 2.3.2.1 is een plaatje van een shuttle te zien.

Figuur 2.3.2.1: Een shuttle die zich op een baan onder de pijpenbaan bevindt

2.3.3 Automatische transportbaan

De shuttles kunnen de rolcontainers over de pijpenbanen alleen maar in de lengte van de kas verplaatsen. Om de rolcontainer ook in de breedte door de kas te kunnen verplaatsen heeft Logiqs B.V. de automatische transportbaan ontwikkeld.

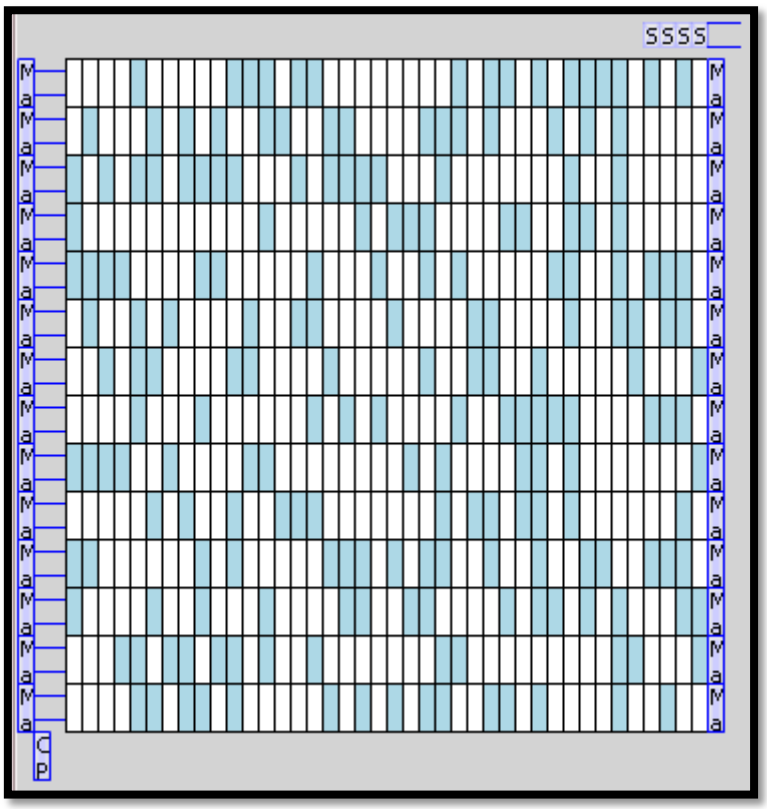
In de kassen liggen meerdere pijpenbanen naast elkaar waar aan ieder uiteinde een automatische transportbaan geïnstalleerd is. Deze transportbaan neemt de rolcontainer over van de shuttle en verplaatst de rolcontainer in de breedte door de kas heen. De transportbanen hebben een verbinding met elkaar waardoor verplaatsing van een rolcontainer vanuit iedere willekeurige pijpenbaan naar een andere willekeurige pijpenbaan mogelijk is. Omdat de automatische transportbaan eigenlijk uit allemaal kleinere machines bestaat en ieder machine een rolcontainer kan vervoeren, kunnen de transportbanen gezamenlijk een zeer grote capaciteit aan rolcontainers tegelijk verplaatsen. Dit is dus een zeer efficiënt transportmiddel in de kas. In figuur 2.3.3.1 is een foto van een automatische transportbaan te zien. De rolcontainer rechts vooraan in het plaatje wordt op het moment dat de foto genomen wordt door de automatische transportbaan verplaatst.

Figuur 2.3.3.1: Transportbaan die rolcontainer aan het verplaatsen is

2.3.4 Dat-A-Control

Dat-A-Control is een door Logiqs B.V. ontwikkeld softwarepakket dat bestaat uit meerdere modules. Iedere module bestaat uit een softwareoplossing voor geautomatiseerde logistieke processen. Dat-A-Control regelt deze processen en geeft inzicht in de teeltgegevens zoals kwaliteit, voorraad en aflevering. Niet iedere kas maakt gebruik van alle modules uit Dat-A-Control. Voor het gebruik van de logistieke systemen van Logiqs B.V. is alleen Dat-A-Logiqs nodig. De andere modules zijn optioneel.

Dat-A-Logiqs laat de tuinder een grafische weergave van zijn kas zien. In deze grafische weergave kan de tuinder door middel van slepen een rolcontainer verplaatsen. Door de rolcontainer van zijn huidige plek naar de plek waar hij de rolcontainer wil hebben staan te slepen wordt de opdracht gegeven. De software is zo ontwikkeld dat de tuinder meerdere opdrachten tegelijk kan ingeven. Naast het geven van opdrachten kan de tuinder ook de gegevens van een individuele rolcontainer opvragen. Zo kan hij precies zien wat er op deze rolcontainer staat en op welke plaatsen deze allemaal geweest is. In figuur 2.3.4.1 is een voorbeeld te zien van een grafische kasweergave.



Figuur 2.3.4.1: Een grafische kas weergave

2.3.5 iCUBE

Naast de automatische transportsystemen in de kassen ontwikkelt Logiqs B.V. de laatste jaren ook systemen voor de opslag van goederen. Een van de laatst ontwikkelde systemen is de iCUBE. De iCUBE is een automatisch palletopslagsysteem.

De iCUBE bestaat uit veel opslagstellingen die allemaal verbonden zijn met elkaar. Een shuttle kan een pallet vanuit iedere locatie in de opslag naar iedere andere locatie in de opslag verplaatsen. De iCUBE zorgt voor een maximaal gebruik van de opslagruimte, zonder ruimte te gebruiken voor looppaden tussen de rijen. In figuur 2.3.5.1 is een foto van een iCUBE in de loods bij Logiqs B.V. te zien.

Figuur 2.3.5.1: De iCUBE in de loods bij Logiqs B.V.

3. Afstudeeropdracht

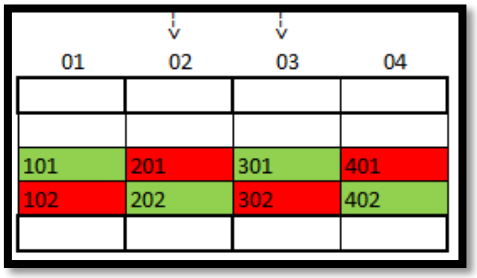
In dit hoofdstuk is de afstudeeropdracht uitgeschreven. Allereerst komt de aanleiding voor het verstrekken van de afstudeeropdracht aan bod. Vervolgens is in paragraaf 3.2 de opdracht omschreven. Paragraaf 3.3 behandeld de deelopdrachten die voortkomen uit de afstudeeropdracht. In de vierde paragraaf bevindt zich de afbakening van de afstudeeropdracht in de laatste paragraaf komen de randvoorwaarden naar voren.

3.1 De aanleiding

Tuinders willen zo goedkoop mogelijk hun producten maken. Daarom vullen de tuinders het hele kasoppervlak met hun producten. Hierdoor gebruiken ze het oppervlak optimaal en worden de producten goedkoper. Het optimaal gebruik van het kasoppervlak heeft ook een nadeel. Zo is er in de kas maar weinig ruimte voor het tijdelijk opslaan van rolcontainers. Dit tijdelijk opslaan is nodig wanneer de orders voor de volgende dag klaargezet worden. Hierbij staan er rolcontainers met orders en rolcontainers zonder orders in dezelfde pijpenbaan. De rolcontainers zonder orders staan dan in de weg en moeten verplaatst worden naar een lege plaats voordat de rolcontainers met orders bereikt kunnen worden. Met een voorbeeld wordt het probleem uitgelegd en zijn de gevolgen duidelijker.

Voorbeeld:
Een tuinder heeft aan het eind van de dag een orderlijst met planten die de volgende dag verzendklaar moeten zijn. Deze planten wil hij de volgende ochtend op één specifieke pijpenbaan in de kas hebben klaarstaan. De planten uit de orderlijst staan op rolcontainers die op willekeurige locaties in de kas staat. Het op de juiste plaats zetten van de rolcontainers heet voorsorteren.

In figuur 3.1.1 is de kasindeling die bij het voorbeeld hoort weergegeven. De kas is in het voorbeeld flink verkleind. De bovenste en onderste rij vakjes stelt een automatische transportbaan voor die de rolcontainers in de breedte verplaatst. Op alle andere locaties in de kas verplaatsen de shuttles de rolcontainers in de lengte. De rode vakjes zijn de rolcontainers met orders. Deze rolcontainers moeten de volgende ochtend in de pijpenbanen staan die met een pijl aangegeven zijn. Deze pijpenbanen worden voorsorteerbanen genoemd. De groene vakjes zijn rolcontainers zonder orders. Ook zijn er ook witte vakjes te zien. Dit zijn lege plaatsen op de pijpenbaan waar rolcontainers tijdelijk opgeslagen kunnen worden. Als laatst staan er nummers in de vakjes en boven de vakjes. De nummers boven de vakjes geven de naam van de pijpenbaan aan. De nummers in de vakjes geven de namen van de rolcontainers aan.



Figuur 3.1.1: Mogelijke kasindeling

Door verschillende verplaatsopdrachten aan het systeem te geven worden de rode rolcontainers van hun plek in de kas naar de juiste pijpenbaan te verplaatsen geeft de tuinder. Het geven van deze opdrachten is het probleem dat de tuinders hebben. Op dit moment moeten de tuinders de opdrachten handmatig in Dat-A-Logiqs invoeren. Het gaat soms om wel meer dan zeshonderd opdrachten. **Het handmatig invoeren van de opdrachten kost de tuinder vaak meer dan een half uur.** Daarnaast is ook de efficiëntie van de gegeven opdrachten ondermaats. Doordat er weinig ruimte in de kas is moeten de tuinders veel groene rolcontainers meerdere keren verplaatsen om de rode rolcontainers op de juiste plek te krijgen. Hierdoor vinden er overbodige verplaatsingen plaats en duurt het verplaatsen van de rolcontainers langer dan nodig. **Soms duurt het zelfs zo lang dat het systeem de volgende ochtend nog steeds bezig is met het uitvoeren van de opdrachten.**

Samengevat zorgt het handmatig invoeren voor twee problemen:

- Het invoeren van de opdrachten kost te veel tijd
- Het uitvoeren van de opdrachten door het systeem kost te veel tijd

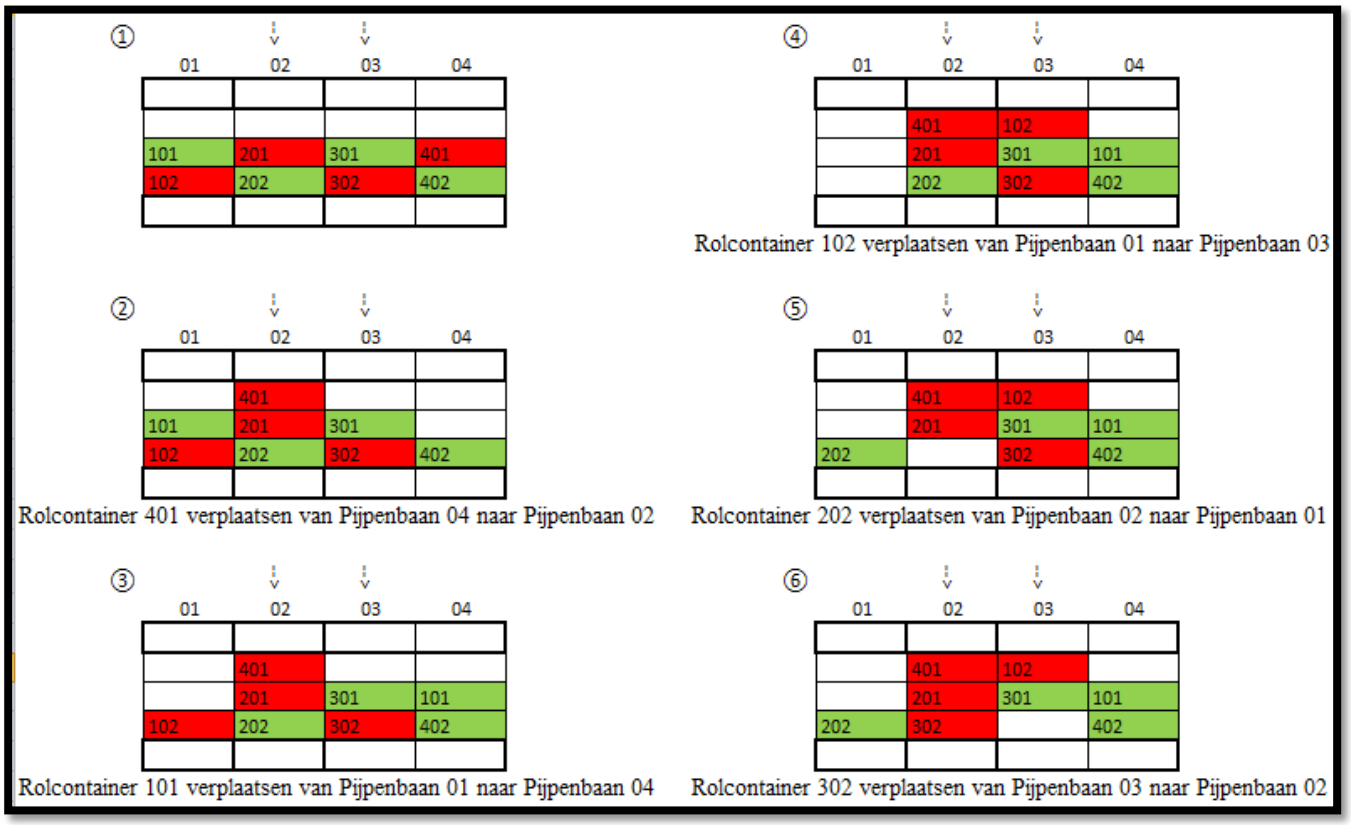
Een van de klanten van Logiqs B.V. die hier tegenaan loopt is het bedrijf Anthura Arndt in Duitsland. Anthura Arndt wil dat het voorsorteren van de rolcontainers in hun dertiende afdeling efficiënter verloopt en heeft Logiqs B.V. gevraagd naar een oplossing.

Naast het helpen van deze klant ziet Logiqs B.V. het voorsorteren als een kans om haar systemen te verbeteren met een nieuwe toepassing. Als laatste kijkt Logiqs B.V. verder dan het voorsorteren in het interne transportsysteem. Logiqs B.V. ziet ook een mogelijkheid om deze zelfde verbetering in de iCUBE te implementeren.

3.2 De opdracht

Het doel van de afstudeerstage is het efficiënter verplaatsen van rolcontainers. De opdracht die hierbij hoort, is het ontwerpen en programmeren van een algoritme dat als output een efficiënte serie opdrachten genereert die de rode rolcontainers voorsorteert. Na het invoeren van de output Dat-A-Logiqs heeft Dat-A-Logiqs een efficiënte manier om de rode rolcontainers voor te sorteren.

De indeling uit figuur 3.1.1 wordt nu uitgewerkt om een beeld te krijgen van de uitkomsten van het algoritme. Deze uitwerking is te zien in figuur 3.2.1. Hier zijn zes kasindelingen weergegeven. De kasindeling met nummer één is de kasindeling uit figuur 3.1.1. De andere kasindelingen worden bereikt door het uitvoeren van meerdere verplaatsingen. Onder iedere kasindeling staat welke verplaatsing gedaan is om vanuit de vorige kasindeling in de huidige kasindeling te komen. In de zesde kasindeling zijn de rode rolcontainers voorgesorteerd en is het doel bereikt.



Figuur 3.2.1: Kasindeling na het voorsorteren

3.3 De deelopdrachten

Om de opdracht gestructureerd uit te voeren is de opdracht gesplitst in deelopdrachten. Iedere deelopdracht heeft een tussenresultaat. Per deelopdracht is beschreven wat de deelopdracht inhoudt. De deelopdrachten zijn:

- Het analyseren van het logistieke systeem van Anthura Arndt
- Literatuuronderzoek doen naar bestaande algoritmen voor dit probleem.
- De gevonden algoritmen vergelijken met het probleem.
- Het ontwerpen van het beginalgoritme.
- Het programmeren van het beginalgoritme.
- Het testen van het beginalgoritme.
- Literatuuronderzoek naar een verbetering van het beginalgoritme.
- Het programmeren van de verbeteringen in het beginalgoritme
- Het testen van het verbeterde algoritme

De deelopdrachten die leiden tot het beginalgoritme worden na elkaar uitgevoerd. Wanneer er eenmaal een beginalgoritme is, wordt er gezocht naar verbeteringen. Deze verbeteringen worden hierna geïmplementeerd en getest. Het zoeken, implementeren en testen van verbeteringen worden meerdere keren achter elkaar herhaald.

3.3.1 Analyseren van het logistieke systeem van Anthura Arndt

Allereerst worden er twee analyses gedaan op het logistieke systeem van Logiqs B.V. Dit zijn de systeemanalyse en de tijdsanalyse. Na de systeemanalyse is duidelijk wat de details van het probleem zijn en kan gezocht worden naar vergelijkbare problemen. De resultaten uit de tijdsanalyse bepalen wat de graadmeter wordt voor de efficiëntie van de serie opdrachten.

3.3.2 Literatuuronderzoek naar bestaande algoritmen

Er wordt in de literatuur gezocht naar algoritmen die een vergelijkbaar probleem als bij Anthura Arndt oplossen. Het zoeken naar bestaande algoritmen in de literatuur voorkomt dat er tijd besteedt wordt aan het ontwikkelen van een nieuw algoritme terwijl er in de literatuur meerdere algoritmen te vinden zijn.

3.3.3 De gevonden algoritmen vergelijken met het probleem

Door de gevonden algoritmen te vergelijken met het probleem van Anthura Arndt wordt voor ieder algoritme bepaald of deze geschikt is om te gebruiken bij het oplossen van het voorsorteerprobleem. Is deze geschikt dan kan deze meegenomen worden in het beginalgoritme.

3.3.4 Het ontwerpen van het beginalgoritme

Het ontwerpen van het beginalgoritme gebeurt door meerdere algoritmen gevonden bij deelopdracht 3.3.3 samen te voegen. Het algoritme dat ontstaat na het samenvoegen heet het beginalgoritme. Dit algoritme moet de eerste oplossing genereren.

3.3.5 Het programmeren van het beginalgoritme

Deze deelopdracht heeft als doel het programmeren van het beginalgoritme. Dit gebeurt in het programma Visual Studio in de taal C#. Het algoritme wordt nog niet geïmplementeerd in de software van Logiqs B.V. Wel wordt er gewerkt met een console application. Deze geeft de in- en output weer.

3.3.6 Het testen van het beginalgoritme

Na het programmeren is het belangrijk dat het beginalgoritme getest wordt op meerdere punten. Van te voren is met de begeleider overlegd wat de criteria zijn bij het genereren van een oplossing. De enige criteria die er is betreft het genereren van de oplossing zelf. Het algoritme moet een oplossing genereren. Er zijn geen criteria opgesteld naar de tijdsduur van het algoritme en de efficiëntie van de serie opdrachten. Ondanks dat er geen criteria zijn voor duur en efficiëntie wordt hier wel op getest. Er is streeftijd en streefefficiëntie opgesteld.

De streeftijd wordt vastgesteld op drie minuten. Dit allereerst omdat drie minuten wachten niet te lang is. Daarnaast kan een test dan na drie minuten afgebroken worden. Dit scheelt veel tijd.

De efficiëntie van een oplossing wordt weergegeven als het verschil in percentages tussen de efficiëntie van de oplossing en de efficiëntie van het optimum. Dit verschil wordt de efficiëntieafwijking genoemd. Het echte doel is een efficiëntieafwijking van nul procent. De optimale oplossing is dan bereikt. Dit is een utopie. Daarnaast is het bepalen van het optimum erg lastig. Daarom wordt er gestreefd naar een efficiëntieafwijking van minder dan twintig procent voor negentig procent van de gevallen.

Er wordt getest op:

- De efficiëntieafwijking van de gegenereerde oplossing: Het streven is dat bij negentig procent van de oplossingen de afwijking ten opzichte van het optimum hooguit twintig procent is.
- De snelheid van het algoritme: Het streven is dat het algoritme binnen drie minuten een oplossing genereert.
- De uitvoerbaarheid van de serie opdrachten: Het algoritme moet een oplossing genereren die ‘visible’ is.

3.3.7 Literatuuronderzoek naar een verbetering

Na het testen blijkt waar de zwakke punten van het algoritme liggen. Tijdens deze deelopdracht wordt er gezocht naar een verbetering die de zwakke punten van het algoritme verbetert.

3.3.8 Het programmeren van verbeteringen in het beginalgoritme

Na het bepalen van geschikte verbeteringen is het tijd om de verbeteringen te implementeren in het beginalgoritme. De uitkomst van de deelopdracht zijn meerdere beginalgoritme waarbij één of meerdere verbeteringen zijn geïmplementeerd. Door niet alle verbeteringen in één beginalgoritme te implementeren ontstaat er een set met algoritmen die met elkaar te vergelijken zijn.

3.3.9 Het testen van het verbeterde algoritme

Het testen van de algoritmen gebeurt in deze deelopdracht. De algoritmen worden op dezelfde punten getest als gebeurt is bij deelopdracht 3.3.6:

- De ‘visibility’ van de oplossing
- De tijd die het kost voor het genereren van een oplossing
- De efficiëntie van het algoritme

3.4 De afbakening van de opdracht

Bij het uitvoeren van een afstudeeropdracht is het belangrijk om van te voren te formuleren wat er wel en wat er niet bij de opdracht hoort. Dit heet de afbakening. In deze paragraaf is de afbakening beschreven. In de volgende drie categorieën wordt de opdracht afgebakend:

- De bestemming van het algoritme
- De efficiëntie van het algoritme
- Het programmeren van het algoritme

3.4.1 De bestemming van het algoritme

Het algoritme moet specifiek werken voor het logistieke systeem van de dertiende afdeling bij Anthura Arndt. Dus niet voor de logistieke systemen bij andere bedrijven. Het is voor Logiqs B.V. wel mogelijk om na afloop van de stage het algoritme aan te passen zodat het bij andere bedrijven ook bruikbaar is.

3.4.2 De efficiëntie van het algoritme

De efficiëntie kan op meerdere manieren gedefinieerd worden. Voor deze opdracht is er gekozen het aantal verplaatsingen als efficiëntie te gebruiken. Een kleiner aantal verplaatsingen resulteert in een lagere efficiëntieafwijking. Bij de tijdsanalyse wordt bepaald hoe een verplaatsing precies gedefinieerd is. Een andere mogelijkheid is het uitrekenen van de duur van de serie opdrachten in seconden. Deze mogelijkheid kost meer tijd om te implementeren en wordt daarom niet doorgevoerd.

3.4.3 Het programmeren van het algoritme

De programmeertaal voor het algoritme is C#. Het algoritme wordt geïmplementeerd in het stuk programmeerwerk dat de verplaatsingen in de kas regelt. Om het programma werkend te krijgen moet er een aangepaste userinterface komen. Het maken van deze userinterface is een taak van Logiqs B.V.

3.4.4 Samenvatting

De volgende taken voert de student uit:

- Een algoritme maken voor de dertiende afdeling van Anthura Arndt.
- De efficiëntieafwijking berekenen op basis van het totaal aantal verplaatsingen.
- Het algoritme programmeren in C#.

De volgende taken voert de student niet uit:

- Algoritme maken voor andere afdelingen van Anthura Arndt of andere bedrijven.
- De efficiëntieafwijking berekenen op basis van tijd.
- De lay-out van het programma programmeren.

3.5 Randvoorwaarden

Bij een aantal deelopdrachten is de hulp van Logiqs B.V. nodig. Deze paragraaf beschrijft bij welke deelopdrachten er hulp nodig is van Logiqs B.V.

Om de opdracht uit te kunnen voeren moet er allereerst de beschikking komen over een aantal programma's. Dit zijn:

- Dat-A-Logiqs voor het analyseren van de kasindeling
- Visual Studio voor het programmeren van het algoritme
- Source Tree voor het online zetten van het algoritme

Voor de analyse is ook de kasindeling zelf nodig. Wanneer Logiqs B.V. deze aanlevert kan het ingeladen worden in Dat-A-Logiqs en kan deze geanalyseerd worden.

Voor het testen van de algoritmen moet er in Dat-A-Logiqs een extra toepassing komen. De extra toepassing moet informatie kunnen laden. Deze informatie betreft de voor te sorteren rolcontainers. Ook moet er een extra toepassing voor de output komen. Deze moet namelijk automatisch Dat-A-Logiqs ingeladen worden.

4. Analyse logistieke systeem

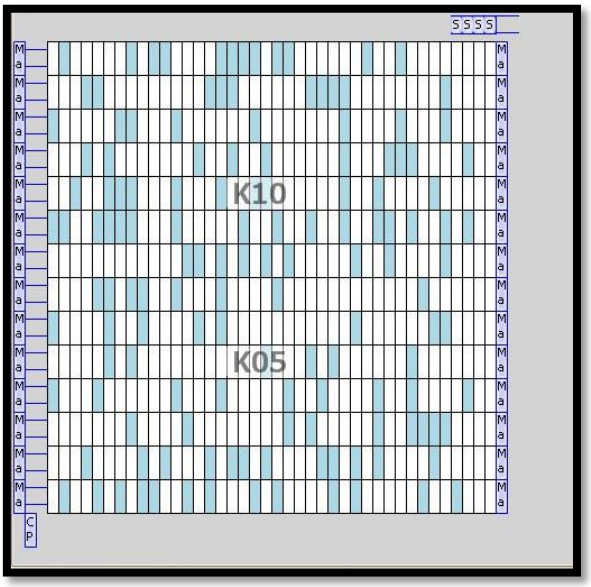
In dit hoofdstuk zijn de uitkomsten van de analyse op het logistieke systeem van de dertiende afdeling van Anthura Arndt beschreven. Deze analyse is gedaan zodat er een volledig beeld van het probleem gecreëerd is.

Er zijn twee analyses uitgevoerd. Dit zijn de systeemanalyse en de tijdsanalyse. Bij de systeemanalyse is de dertiende afdeling van Anthura Arndt geanalyseerd. Hierbij is bekeken hoe de afdeling er uitziet en hoe de logistiek er precies werkt. Bij de tijdsanalyse is de gemiddelde duur van de verschillende opdrachten bepaald. Dit is gedaan zodat bepaalt kan worden wat de graadmeter wordt voor de efficiëntieafwijking van een later uitkomst.

4.1 Systeemanalyse

Bij de systeemanalyse is alleen naar de dertiende afdeling van Anthura Arndt gekeken. Deze afdeling bevat veertien pijpenbanen die naast elkaar liggen. Op iedere pijpenbaan is ruimte voor 42 rolcontainers. De pijpenbanen zijn niet volledig gevuld. Op iedere pijpenbaan zijn veertig rolcontainers geplaatst. Er zijn twee lege plaatsen per pijpenbaan.

Aan beide uiteinden van de pijpenbanen bevinden zich transportbanen. In figuur 4.1.1 is het kasoverzicht van afdeling dertien te zien. In het overzicht liggen de pijpenbanen horizontaal, de transportbanen liggen verticaal. In het overzicht zijn de rolcontainers getekend als rechthoekjes. De grijze rechthoekjes zijn de rolcontainers met orders. Deze moeten voorgesorteerd worden.

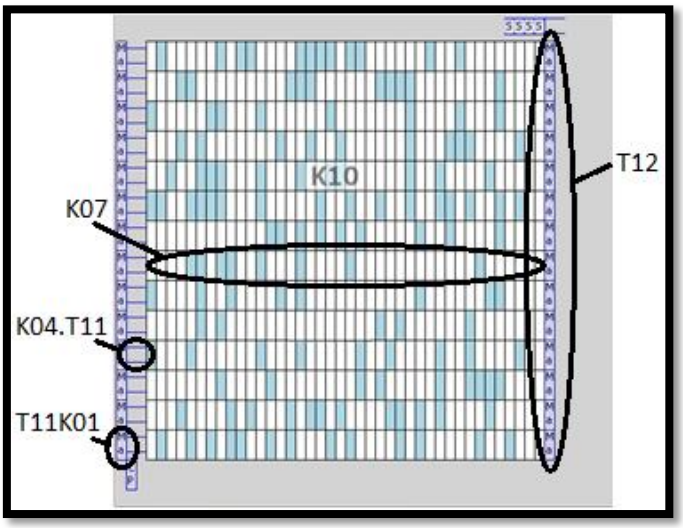


Figuur 4.1.1: Kasoverzicht van de dertiende afdeling

Onder de pijpenbanen rijden shuttles. Door het gebruik van shuttles kunnen meerdere opdrachten tegelijk uitgevoerd worden. In de dertiende afdeling van Anthura Arndt zijn nog geen shuttles aanwezig. Daarom is besloten de onderdelen die afhankelijk zijn van het aantal shuttles niet mee te nemen aan de graadmeter voor de efficiëntieafwijking.

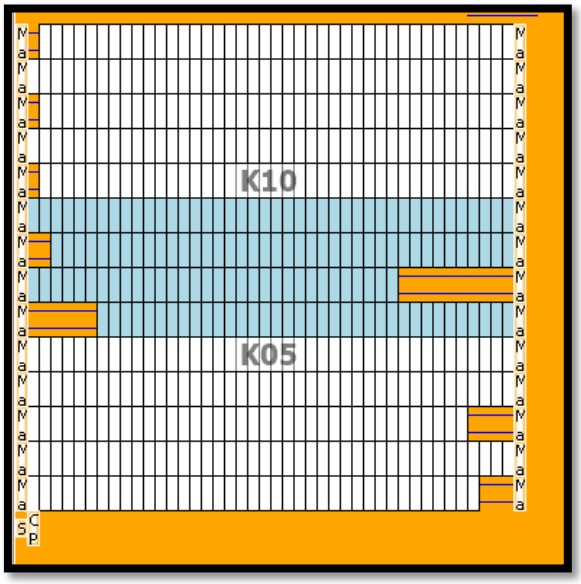
Alles wat in het kasoverzicht aanwezig is heeft een naam. Zo zijn de pijpenbanen genummerd van K01 tot K14. In figuur 4.1.2 is pijpenbaan K01 de onderste, en K14 de bovenste. De linker transportbaan heet T11 en de rechter transportbaan heet T12. Eerder is beschreven dat de transportbanen eigenlijk veertien kleinere aan elkaar gekoppelde machines zijn. Deze machines hebben allemaal een aparte naam. De naam voor zo'n machine bestaat uit de naam van de transportbaan + de naam van de pijpenbaan die eraan grenst. Voorbeeld: Een machine die onderdeel is van de transportbaan T11 en aan de pijpenbaan K01 grenst heet T11K01.

Ook heeft iedere pijpenbaan een speciale locatie. De speciale locatie is vanuit de transportbaan gezien de eerste plek op de pijpenbaan waar een rolcontainer kan staan. Bij Anthura Arndt is een pijpenbaan verbonden met twee transportbanen en zijn er twee van deze locaties. Voorbeeld: de speciale locatie op de pijpenbaan K04 die grenst aan de transportbaan T11 heet K04.T11. In figuur 4.1.2 is dezelfde kas te zien als in figuur 4.1.1. In het figuur zijn bij een aantal machines de benaming toegevoegd.



Figuur 4.1.2: Kasoverzicht met benamingen

Eerder in deze paragaaf is beschreven dat de grijze rolcontainers orders bevatten voor de volgende ochtend. Deze moeten naar een specifieke locatie voorgesorteerd worden. Deze specifieke locatie betreft de pijpenbanen K03 tot K09. Deze pijpenbanen heten de voorsorteerbanen. De rolcontainers moeten vooraan op de pijpenbaan komen te staan. Met vooraan staan wordt bedoeld dat de rolcontainers via T11 de baan kunnen verlaten zonder dat er een witte rolcontainer in de weg staat. In figuur 4.1.3 is een overzicht van de dertiende afdeling weergegeven waarin de grijze rolcontainers in de pijpenbanen K06 tot en met K09 zijn voorgesorteerd.



Figuur 4.1.3: Kasoverzicht na het voorsorteren

In het figuur is te zien dat de K03, K04 en K05 geen grijze rolcontainers bevatten terwijl het wel voorsorteerbanen zijn. Dit komt omdat het geen eis is dat de rolcontainers gelijk over de voorsorteerbanen verdeeld moeten worden. In dit voorbeeld is ervoor gekozen om zo weinig mogelijk pijpenbanen te gebruiken bij het voorsorteren.

Nu bekend is hoe het systeem eruit ziet kan gedefinieerd worden wat een verplaatsing is. Een verplaatsing bestaat uit meerdere instructies. Allereerst moet de rolcontainer vanuit de pijpenbaan naar de transportbaan gebracht worden. Deze opdracht heet de shuttle-instructie (UIT) en wordt door een shuttle uitgevoerd. Hierna wordt de rolcontainer over de transportbaan verplaatst richting de gewenste pijpenbaan. Omdat de transportbaan uit aparte machines bestaat kunnen voor deze verplaatsing meerdere instructies nodig zijn. Dit zijn de transportbaaninstructies. Als laatste wordt de rolcontainer uit de transportbaan de pijpenbaan in geduwd. Ook deze verplaatsing wordt door een shuttle gedaan. De naam voor deze instructie is de shuttle-instructie (IN).

4.2 Tijdsanalyse

Door een tijdsanalyse te doen is onderzocht welk onderdeel van het systeem het grootste effect heeft op de duur van de opdrachten. Dit is belangrijk bij het definiëren van een verplaatsing. Deze verplaatsing is onderdeel van de graadmeter die de efficiëntieafwijking van de oplossing bepaald.

Een werknemer van Anthura Arndt heeft drie filmpjes gemaakt van verschillende verplaatsingen in de kas. Op deze filmpjes zijn de instructies te zien die in de vorige paragraaf beschreven zijn. Aan de hand van de filmpjes is voor iedere instructie bepaald wat de tijdsduur ervan is. De tijdsduren van de drie instructies uit het filmpje zijn:

- Transportbaaninstructie: 16 seconden
- Shuttle-instructie (IN): 25 seconden
- Shuttle-instructie (UIT): 66 seconden

Conclusie:

Uit deze analyse komt naar voren dat de verplaatsing waarbij de shuttle een rolcontainer uit de pijpenbaan haalt veruit het langst duurt. Deze verplaatsing heeft dan ook de meeste invloed op de uiteindelijke eindtijd. Doordat iedere verplaatsing een shuttleopdracht (UIT) heeft is het totaal aantal verplaatsingen gelijk aan het aantal shuttleopdrachten (UIT). De graadmeter voor de efficiëntieafwijking wordt daarom vastgesteld op het totaal aantal verplaatsingen.

5. Literatuuronderzoek

Er is in de literatuur gezocht naar bestaande algoritmen die het voorsorteerprobleem van Anthura Arndt op kunnen lossen. In dit hoofdstuk zijn de bevindingen van dit literatuuronderzoek beschreven.

Eerst is beschreven tot welke klasse het probleem van Anthura Arndt behoort. Hierna zijn er problemen beschreven die overeenkomen met het probleem van Anthura Arndt. Als laatste worden meerdere oplosmethoden besproken.

5.1 Probleemklasse

Bij het probleem van Anthura Arndt is er een begin- en een eindsituatie. De eindsituatie wordt bereikt door vanuit de beginsituatie een serie verplaatsingen achter elkaar uit te voeren. De verplaatsingen in deze serie en de volgorde ervan hangt af van de keuzes die tijdens het opstellen van de serie gemaakt zijn. Het maken van een keuze heeft namelijk invloed op de vervolgopties.

Een probleem waarbij de uitkomst afhankelijk is van de keuzes die gemaakt zijn tijdens het proces wordt een beslissingsprobleem genoemd. Het probleem van Anthura Arndt is zo'n beslissingsprobleem.

Naast het definiëren van het probleem kan het probleem ook in een klasse geplaatst worden. In een rapport van de universiteit van Brussel over beslissingsproblemen staat dat er een klasse bestaat waarbij de optimale oplossing bereikt wordt door alle mogelijke situaties te onderzoeken. Deze klasse wordt de NP-volledige klasse genoemd en hier valt ook het probleem van Anthura Arndt in. Een aantal voorbeelden van NP-volledige problemen zijn:

- Het handelsreizigersprobleem
- Het knapzakprobleem
- Het vervulbaarheidsprobleem

Conclusie:
Het probleem van Anthura Arndt valt in de NP-volledige klasse omdat het allereerst een beslissingsprobleem. Daarnaast wordt de optimale oplossing pas gevonden wanneer alle mogelijkheden doorzocht zijn. Dit zijn ook de kenmerken van een NP-volledig probleem.

In de volgende paragrafen zijn problemen die ook in de NP-volledige klasse vallen en een duidelijke overeenkomst hebben met het probleem van Anthura Arndt beschreven.

5.2 Overeenkomstige problemen

Er zijn meerdere NP-volledig problemen te vinden in de literatuur. Van deze problemen zijn er twee die grote overeenkomsten met het probleem van Anthura Arndt hebben. Deze paragraaf beschrijft deze problemen.

Allereerst komt het spel FreeCell naar voren. De informatie over dit spel komt uit een YouTube-filmpje waarin Léon Bouquiet uitlegt hoe hij het spel opgelost met behulp van artificial intelligence (kunstmatige intelligentie). Het tweede probleem betreft het spel genaamd ‘de schuifpuzzel’. Ook hier is sprak van een NP-volledig probleem. In een rapport van de Vrije Universiteit Brussel is weergegeven hoe de schuifpuzzel via een algoritme op basis van artificial intelligence op te lossen is.

5.2.1 FreeCell

FreeCell is een kaartspel waarbij de beginsituatie bestaat uit acht stapeltjes met kaarten, vier lege cellen (de FreeCells) en een plaats waar alle kaarten uiteindelijk moeten komen. Dit zijn de zogeheten foundations. Een van de mogelijke beginsituaties is weergegeven in figuur 5.2.1.1.

Figuur 5.2.1.1: De beginsituatie van een FreeCell-spel

De speler kan een kaart verplaatsen door deze op een andere kaart op een stapel te leggen. Hierbij moeten de kleuren van de twee kaarten verschillend zijn. Daarnaast moet de betreffende kaart één waarde lager zijn dan de waarde van de kaart die er al lag. Ook kan een willekeurige kaart verplaatst worden naar één van de vier vrije cellen (FreeCells) linksboven in figuur 5.2.1.1. De enige criteria voor deze verplaatsing is het hebben van een lege vrije cel. Als laatste kan er een kaart verplaatst worden naar de foundations. Dit kan alleen als er op de foundations een kaart ligt met dezelfde kleur waarvan de waarde op de kaart eentje lager is dan te verplaatsen kaart.

Om een duidelijker beeld te krijgen over de mogelijke verplaatsingen die er zijn is de beginsituatie in figuur 5.2.1.1 gedeeltelijk uitgewerkt. In figuur 5.2.1.2 is dezelfde beginsituatie te zien. Nu zijn er drie mogelijke verplaatsingen aangegeven met zwarte pijlen. Dit zijn:

- Een verplaatsing van een stapels naar een foundation (Klaveren aas)
- Een verplaatsing van een stapel naar een vrije cel (Klaveren koning)
- Een verplaatsing van een stapel naar een stapel (Schoppen zeven)

Figuur 5.2.1.2: De beginsituatie met ingetekende verplaatsingen

Het spel is klaar wanneer alle kaarten opgestapeld op de foundations liggen. De onderste kaart is een aas en de bovenste een koning. De eindoplossing is in figuur 5.2.1.3 weergegeven.

Figuur 5.2.1.3: De eindoplossing van een FreeCell-spel

5.2.2 FreeCell in vergelijking met Anthura

Bij het vergelijken van het spel FreeCell met het probleem van Anthura Arndt zijn er verschillen en overeenkomsten. Beide zijn in deze paragraaf beschreven. De verschillen en overeenkomsten behoren tot bepaalde categorieën. De categorieën die naar voren komen zijn:

- Probleemklasse
- Stapels
- Objecten
- Automatische transportbaan
- Regels

Probleemklasse:

De eerste overeenkomst betreft het soort probleem. Beide problemen vallen in de klasse van de NP-volledige problemen. De zekerheid over het optimum is er alleen als alle opties zijn nagegaan. Daarnaast geldt ook dat de eindoplossing bij beide problemen te vinden is door meerdere stappen achtereenvolgens uit te voeren. Een stap heeft altijd gevolgen voor de stappen die daarna komen.

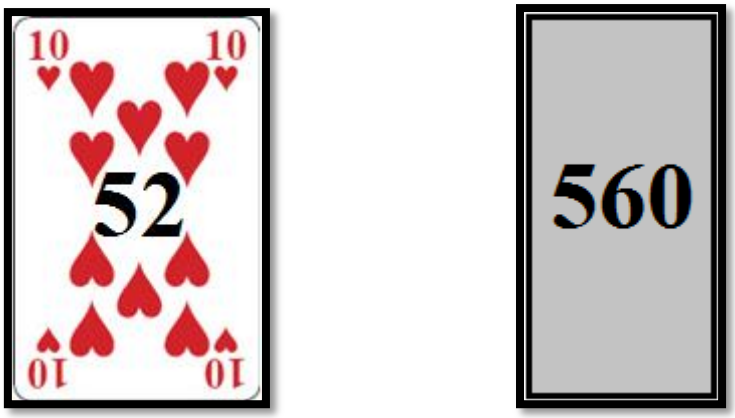
Stapels:

Een andere overeenkomst betreft de stapels in FreeCell. Ook Anthura Arndt heeft stapels. Deze heten alleen geen stapels maar pijpenbanen. Naast een overeenkomst is er ook een verschil. FreeCell heeft acht stapels terwijl Anthura Arndt er veertien heeft.

Doorgaand op de pijpenbanen is er nog een overeenkomst. Zo zijn de voorsorteerbanen vergelijkbaar met de foundations in FreeCell. Bij beide is er van te voren bepaald wat de eindlocatie van de objecten is. Ook hier is er een verschil. Anthura Arndt heeft zeven pijpenbanen terwijl FreeCell maar vier foundations heeft. Daarnaast maakt de volgorde van rolcontainers bij Anthura Arndt niet uit en is dit wel van belang bij FreeCell.

Objecten:

Een ander verschil zit hem in de objecten. Zo heeft Anthura Arndt rolcontainers en FreeCell kaarten. Daarnaast is ook het aantal objecten anders. FreeCell heeft 52 kaarten terwijl Anthura Arndt 560 rolcontainers heeft.



Figuur 5.2.2.1: Links de kaart, rechts de rolcontainer

Automatische transportbaan:

Een volgend verschil heeft te maken met de automatische transportbaan. Anthura Arndt heeft aan beide kanten van de pijpenbaan een automatische transportbaan. Hierdoor kan de rolcontainer de pijpenbaan aan beide kanten verlaten. Bij FreeCell kan een kaart alleen vanaf de voorkant de stapel verlaten, niet vanaf de achterkant.

Regels:

Als laatste zijn er in FreeCell meerdere regels voor het verplaatsen van een kaart. Zo is het niet toegestaan een zwarte kaart op een andere zwarte kaart te leggen. Hetzelfde geldt voor een rode kaart op een rode kaart. Ook moet een kaart die naar een stapel verplaatst wordt één waarde lager zijn dan de kaart die al op de stapel ligt. Bij Anthura Arndt gelden er maar twee regels voor de verplaatsingen. Dit zijn:

- Verplaatsen kan alleen naar een pijpenbaan met een lege plaats
- Als een rolcontainer aan de bovenkant een pijpenbaan verlaat moet deze er bij een andere pijpenbaan aan de bovenkant in. Hetzelfde geldt voor de onderkant.

Samenvatting:

De overeenkomsten tussen Anthura Arndt en FreeCell zijn:

- Beide problemen vallen in de klasse NP-volledig en zijn op te lossen door alle opties na te gaan.
- Voor beide problemen geldt dat de eindsituatie bereikt wordt door achtereenvolgens meerdere stappen uit te voeren.
- De voorsorteerbanen zijn vergelijkbaar met de foundations in FreeCell.
- De stapels van FreeCell zijn vergelijkbaar met de pijpenbanen in Anthura Arndt.

De verschillen tussen Anthura Arndt en FreeCell zijn:

- Anthura Arndt heeft veertien pijpenbanen terwijl FreeCell maar acht stapels heeft.
- De volgorde van de rolcontainers in de voorsorteerbanen maakt niet uit. Bij FreeCell moeten de kaarten van aas tot koning komen te liggen.
- FreeCell heeft kaarten, Anthura Arndt rolcontainers.
- Bij Anthura Arndt kunnen de rolcontainers aan twee kanten de pijpenbaan verlaten. Bij FreeCell kan de kaart maar aan één kant de stapel verlaten.
- Anthura Arndt heeft 560 rolcontainers terwijl FreeCell maar 52 kaarten heeft
- In FreeCell zijn er een aantal regels dat het aantal opties per situatie verkleint. In afdeling dertien zijn dit er minder.

Conclusie:

Uit de vergelijking kan geconcludeerd worden dat er zowel overeenkomsten als verschillen zijn tussen het probleem van Anthura Arndt en FreeCell. De onderdelen die overeenkomen zijn bij beide problemen nodig voor het oplossen van het probleem. De onderdelen die verschillen geven aan wat de omvang van het probleem precies is. Dat hier verschillen in zijn veranderd niets aan de oplosmethode.

5.2.3 Schuifpuzzel

Een ander spel dat overeenkomsten heeft met het probleem van Anthura Arndt is de schuifpuzzel. De schuifpuzzel is een spel op een bord met vierkante vakjes. Het bord is vier bij vier groot en heeft zestien vakjes. Van deze zestien vakjes zijn er vijftien gevuld met een tegeltje. Het zestiende vakje is leeg en wordt gebruikt om de tegeltje tijdelijk naar toe te verschuiven. In figuur 5.2.3.1 is de beginsituatie van een schuifpuzzel te zien.

Figuur 5.2.3.1: De beginsituatie van een schuifpuzzel

Door het lege vakje te gebruiken kan een tegeltje verplaatst worden. In de afbeelding hierboven is het mogelijk de tegeltjes met twaalf en vijftien erop te verschuiven. Door één van deze twee tegeltjes te verschuiven komt er een ander vakje leeg te liggen. Dit lege vakje levert nieuwe opties op. Dit wordt verduidelijkt met een voorbeeld.

Voorbeeld: Door het tegeltje met twaalf erop te verschuiven komt er een nieuw vakje vrij. De uitkomst van de verschuiving van het tegeltje met twaalf erop is weergegeven in figuur 5.2.3.2. In de nieuwe situatie is er een ander vakje leeg. Dit vakje kan door de tegeltjes met acht, twaalf en dertien gebruikt worden om naartoe te verschuiven.

Figuur 5.2.3.2: De uitkomst na de verschuiving van het tegeltje met twaalf

Door steeds opnieuw tegeltjes te verschuiven naar een leeg vakje ontstaan er constant nieuwe situaties. Het verschuiven van de tegeltjes gebeurt tot het spel uitgespeeld is. Het spel is uitgespeeld als de situatie in figuur 5.2.3.3 bereikt is.

Figuur 5.2.3.3: De eindsituatie van de schuifpuzzel

In deze paragraaf is gesproken over een schuifpuzzel van vier bij vier met cijfers in de vakjes. Er zijn ook grotere en kleinere schuifpuzzels. Daarnaast kunnen de cijfers ook vervangen worden door een plaatje. Het spelletje werkt bij alle versies hetzelfde.

5.2.4 De schuifpuzzel in vergelijking met Anthura

Bij het vergelijken van de schuifpuzzel met het probleem bij Anthura Arndt komen er zowel overeenkomsten als verschillen naar voren. De overeenkomsten en verschillen vallen in bepaalde categorieën. Per categorie zijn de overeenkomsten en verschillen beschreven. De categorieën zijn:

- Probleemklasse
- Objecten
- Opslagplaatsen
- Regels

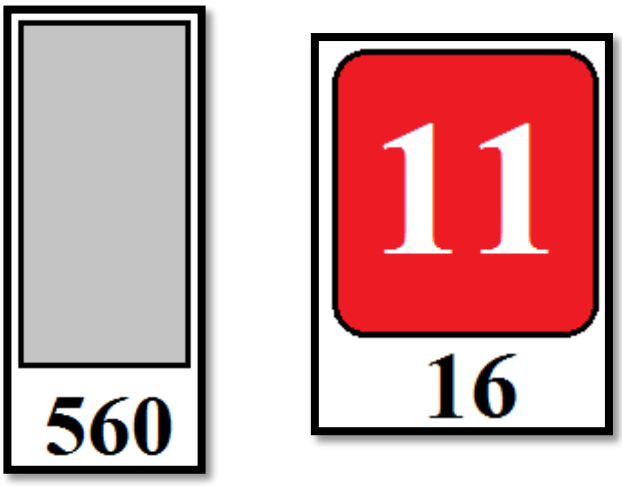
Probleemklasse:

Voor beide problemen geldt dat ze vallen in de klasse NP-volledige problemen. Dit betekent dat de optimale oplossing alleen met zekerheid gevonden is wanneer alle opties nagegaan zijn.

Een andere overeenkomst is het feit dat de oplossing bereikt wordt door meerdere stappen achter elkaar uit te voeren. Bij de schuifpuzzel zijn dit de verschuivingen van de tegeltjes. Bij Anthura Arndt zijn dit de verplaatsingen van de rolcontainers.

Objecten:

Het eerste verschil is het verschil in objecten. Bij de schuifpuzzel zijn het tegeltjes die verschuiven. Bij Anthura Arndt gaat het om rolcontainers. Daarnaast is ook het aantal objecten verschillend. Een schuifpuzzel heeft meestal zestien tegeltjes terwijl Anthura Arndt beschikt over 560 rolcontainers.



Figuur 5.2.4.1: Links de rolcontainer, rechts het tegeltje

Opslagplaatsen:

Een ander verschil is het aantal lege plaatsen. Bij de schuifpuzzel is maar één leeg vakje. Bij Anthura Arndt zijn er 28 lege plaatsen. Iedere lege plaats zorgt ervoor dat een verplaatsing mogelijk is. Zijn er weinig lege plaatsen dan zijn er ook weinig verplaatsingen mogelijk. In dit geval zijn er bij Anthura Arndt daarom meer verplaatsingen mogelijk.

Regels:

Als laatste is het een regel dat de tegeltjes bij de schuifpuzzel in een vaststaande volgorde moeten staan voordat het spel uitgespeeld is. Bij Anthura Arndt isde volgorde van de rolcontainers op de voorsorteerbaan niet belangrijk.

Samenvatting:
De overeenkomsten tussen Anthura Arndt en de schuifpuzzel:

- Beide problemen vallen in de klasse NP-volledig en zijn dus op te lossen door alle opties na te gaan
- Voor beide problemen geldt dat de oplossing bereikt is na het achtereenvolgens uitvoeren van meerdere stappen

De verschillen tussen Anthura Arndt en de schuifpuzzel:

- Anthura Arndt heeft rolcontainers. De schuifpuzzel heeft tegeltjes.
- De schuifpuzzel heeft één leeg plekje. Anthura Arndt heeft 28 lege plaatsen.
- Anthura Arndt heeft 560 rolcontainers terwijl de schuifpuzzel meestal maar zestien tegeltjes heeft.
- De volgorde van de rolcontainers maakt niet uit. Bij de schuifpuzzel is er wel een volgorde van tegeltjes

Conclusie:
Uit de vergelijking kan geconcludeerd worden dat er zowel overeenkomsten als verschillen zijn tussen het probleem van Anthura Arndt en de schuifpuzzel. Ook hier geldt dat de onderdelen die overeenkomen bij beide problemen nodig zijn voor het oplossen van het probleem. De onderdelen die verschillen geven alleen de omvang van het probleem aan. De verschillen veranderden dus niets aan de oplosmethode.

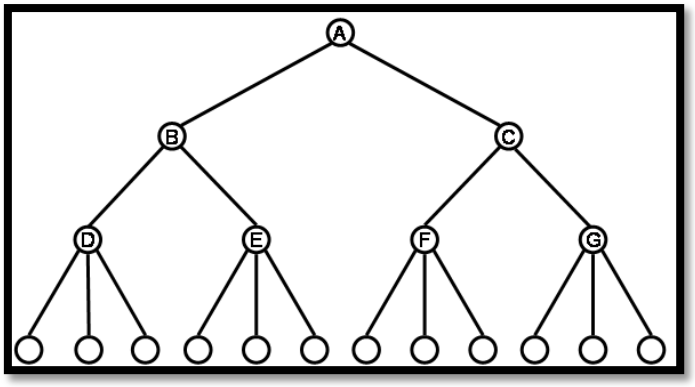
5.2.5 Conclusie

Uit bovenstaande paragrafen is op de maken dat FreeCell en de schuifpuzzel dezelfde soort spellen zijn. De overeenkomsten die FreeCell heeft met het probleem lijken heel erg op de overeenkomsten die de schuifpuzzel heeft met FreeCell. Dit geldt ook voor de verschillen. Daarnaast zijn het verschillen die het probleem alleen in omvang veranderen. De oplosmethode blijft wel hetzelfde. Op basis hiervan kan gezegd worden dat niet het soort spel belangrijk is maar de manier waarop deze opgelost wordt. Het maakt daarom niet uit welk spel er als basis genomen wordt voor het oplossen van het probleem van Anthura Arndt.

5.3 Verschillende oplosmethoden

In paragraaf 5.1 kwam naar voren dat de optimale oplossing bereikt wordt wanneer alle mogelijke situatie nagegaan worden. Het nagaan van alle situatie wordt de brute-force search genoemd. De brute-force search is weer te geven in een zoekboom. Deze paragraaf beschrijft via een zoekboom hoe de optimale oplossing gevonden wordt.

Allereerst wordt er een zoekboom als voorbeeld opgesteld. De beginsituatie krijgt de letter A. In deze beginsituatie zijn er twee opties. De ene leidt tot de nieuwe situatie B en de andere leidt tot situatie C. Deze nieuwe situaties hebben ook twee opties. De opties in situatie B leiden tot de nieuwe situaties D en E. De opties in situatie C leiden tot de nieuwe situaties F en G. Bij de derde keuze zijn er telkens drie opties. Deze zijn verder niet uitgewerkt en hebben daarom geen letter. In figuur 5.3.1 is de zoekboom weergegeven. De cirkels met letters stellen situaties voor. De lijntjes geven de opties weer die leiden tot de volgende situatie.



Figuur 5.3.1: De zoekboom

In de zoekboom is te zien dat het aantal situaties per ‘laag’ snel groeit. De eerste laag bevat alleen de startsituatie. In de tweede laag bevinden zich twee situaties (B en C). Laag drie heeft vier situaties (D, E, F en G) en laag vier heeft al twaalf situaties.

Dit is het probleem van de Brute-force search. Het aantal situaties, en daarmee ook het aantal opties, neemt exponentieel toe. Het nagaan van alle opties wordt hierdoor al snel te groot voor een computer. Het is hierom erg belangrijk dat er op een handige manier naar een oplossing in de zoekboom gezocht wordt.

Er zijn meerdere manieren om uit de zoekboom een oplossing te halen. In de volgende paragrafen zijn vier manieren beschreven. Dit zijn:

- De Depth-first traversal
- De Breadth-first traversal
- De Greedy search
- De Best-first search

De theorie van de Depth-first traversal, de Breadth-first traversal en de Greedy search komen uit een rapport van de Universiteit van Brussel. Het hoofdstuk dat deze drie zoekstrategieën beschrijft heet: “Hoofdstuk 9 Beslissingsalgoritmen”. De Best-first search is uitgelegd in het filmpje van Léon Bouquiet.

5.3.1 Depth-first traversal

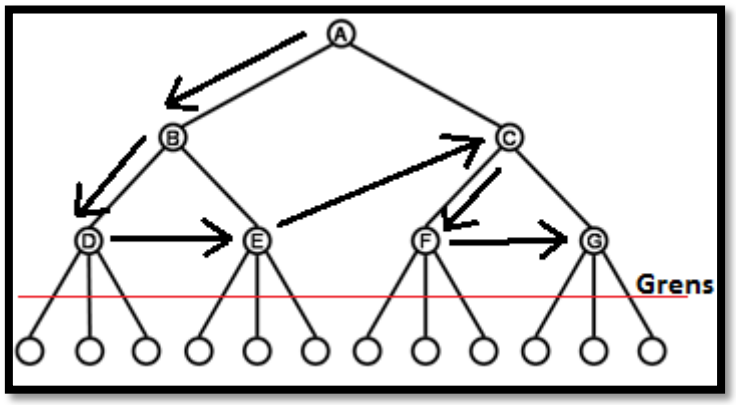
Bij de Depth-first traversal wordt er in de diepte van de boom gezocht naar een oplossing. Bij iedere beslissing krijgt de linker optie in de boom de voorkeur. Het zoeken in de diepte gebeurt tot er een bepaalde grens bereikt is. Deze grens wordt van te voren bepaald. Na het bereiken van de grens wordt in de tak teruggegaan naar de eerstvolgende situatie waarin er een optie is die nog niet gekozen is.

Door middel van een voorbeeld wordt het allemaal wat duidelijker. Het voorbeeld gebruikt de boom uit figuur 5.3.1. In dit voorbeeld is de grens vastgesteld op twee verplaatsingen. Het zoeken stopt wanneer de situaties D, E, F of G bereikt zijn.

Uitwerking:

Bij de eerste situatie krijgt de linker optie de voorkeur. Dit is de optie die leidt tot situatie B. In deze situatie kan er gekozen worden tussen opties die leiden tot de situaties D en E. Omdat de optie tot situatie D de linker optie is krijgt deze de voorkeur. Na deze keuze is de gestelde grens bereikt. Er wordt nu in de tak teruggegaan totdat er een situatie bereikt is een nog niet onderzocht optie heeft. De eerste situatie waarin er een andere optie is, is situatie B. Hierin werd gekozen voor de optie die leidde tot situatie D maar kan er ook voor de optie die leidt tot situatie E gekozen worden. Door nu voor deze optie te kiezen wordt er een andere tak van de boom ingeslagen. Door deze stappen te herhalen wordt er stapsgewijs gezocht naar een oplossing.

Het uitvoeren van de Depth-first traversal op de boom in figuur 5.3.1 levert de volgende volgorde van zoeken op: A-B-D-E-C-F-G. In figuur 5.3.1.1 is de zoekboom te zien met daarin de grens en de route aangegeven met pijlen.



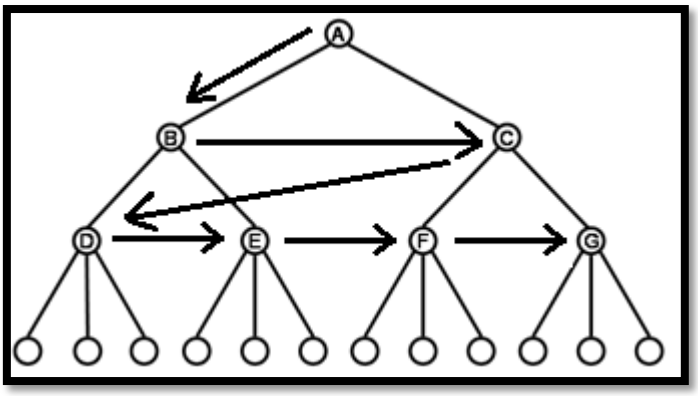
Figuur 5.3.1.1: Zoekboom met grens en route

De Depth-first traversal heeft als voordeel dat er gezocht wordt in de zoekboom zonder deze helemaal op te stellen. Er wordt telkens maar één situatie uitgewerkt. Een nadeel van deze methode is het feit dat de eerste oplossing die gevonden wordt vaak niet de beste oplossing is. Daarnaast kost deze zoek manier veel tijd.

5.3.2 Breadth-first traversal

De Breadth-first traversal zoekt in de breedte van de boom. Per laag wordt bekeken of de situaties in de laag gelijk zijn aan de eindsituatie. In het voorbeeld hieronder is de Breadth-first traversal toegepast op de boom uit figuur 5.3.1.

In de boom van figuur 5.3.1 bevinden zich drie lagen. De eerste laag bevat alleen de situatie A. De tweede laag bevat de situaties B en C. En de laatste laag bevat de situaties D, E, F en G. Bij de Breadth-first traversal wordt eerst voor situatie A bekeken of dit de oplossing is. Hierna wordt dit voor de situaties B en C gedaan. Als laatste zijn de situaties D, E, F en G aan de beurt. In figuur 5.3.2.1 is de zoekboom te zien horend bij deze zoekstrategie. De pijlen geven de route aan waarin gezocht wordt.

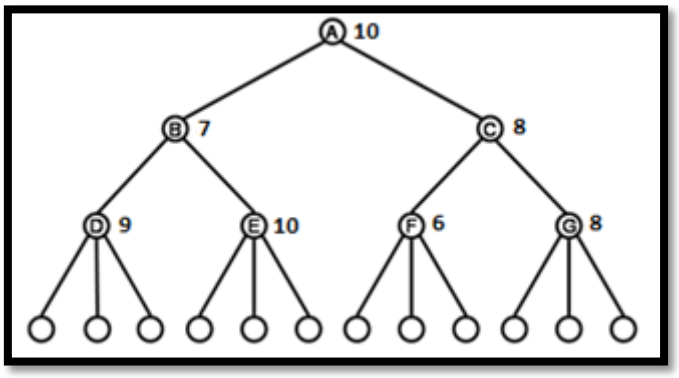


Figuur 5.3.2.1: Zoekboom met pijlen die de route aangeven

De Breadth-first traversal is een inefficiënte manier van zoeken. Eerst moet de hele laag met situaties aangemaakt worden voordat deze onderzocht kan worden. Eerder is ook beschreven dat het aantal situaties per laag exponentieel toeneemt. De tijd voor het doorzoeken van een laag stijgt ook exponentieel. Naast nadelen heeft deze zoekstrategie ook voordelen. De eerste oplossing die gevonden wordt is namelijk gelijk de optimale oplossing.

5.3.3 Greedy search

Bij de Greedy search wordt er gericht gezocht in de boom. Het kiezen van de volgende situatie gebeurt aan de hand van een cijfer. Dit cijfer geeft aan hoe dicht de situatie bij de oplossing ligt. Een laag cijfer betekent dat de situatie dicht bij de oplossing ligt. Een hoog cijfer betekent dat de situatie nog ver van de oplossing verwijderd is. Door constant de situatie met het laagste cijfer te kiezen wordt er verder gegaan met de situatie die op dat moment de meeste potentie heeft om tot een oplossing te komen. Hierdoor wordt sneller naar een oplossing toe gewerkt. In figuur 5.3.3.1 is dezelfde boom te zien als in figuur 5.3.1. Echter zijn er nu cijfers aan toegevoegd.



Figuur 5.3.3.1: Een zoekboom met cijfers

Wanneer de Greedy search wordt toegepast op de boom van figuur 5.3.3.1 is de volgorde van zoeken als volgt: De eerste situatie is situatie A. In deze situatie zijn er twee opties. Beide opties leiden tot een nieuwe situatie (B en C). Voor deze nieuwe situaties wordt het cijfer bepaald. Het cijfer van situatie B is het laagste en krijgt daarom de voorkeur boven situatie C. Na de keuze wordt situatie C verwijdert. Hierna wordt de gekozen situatie uitgewerkt en voor alle nieuwe situaties bepaald wat de cijfers zijn. De opties in situatie B leiden tot de situatie D en E. Deze hebben respectievelijk de cijfers negen en tien. Omdat situatie D een lager cijfer heeft krijgt de optie die leidt tot situatie D de voorkeur. In dit voorbeeld is er dus in de volgende volgorde door de zoekboom heengegaan: A-B-D.

Bij de Greedy search worden de situatie die niet gekozen worden verwijdert. Dit zorgt ervoor dat veel situaties niet aangemaakt worden. Het voordeel hiervan is dat het veel rekenwerk scheelt. De computer is sneller klaar. Een nadeel van de Greedy search is goed te zien in figuur 5.3.3.1. In de boom is de situatie met het laagste cijfer situatie F. Maar door eerder voor de situatie B te kiezen en situatie C te verwijderen komt het algoritme nooit bij situatie F. En dit terwijl het wel de situatie met het laagste cijfer is. De Greedy search zorgt dus voor een sneller proces van zoeken maar bereikt een goede situatie niet altijd.

5.3.4 Best-first search

Anders dan bij de Greedy search onthoudt de Best-first search een situatie als deze niet gekozen wordt. Bij de Best-first search wordt er gekeken naar de beste onthouden situatie. Het bepalen van de beste situatie gebeurt net als bij de Greedy search met een cijfer.

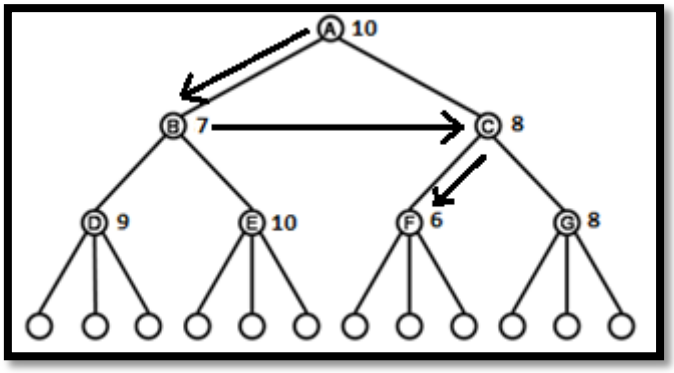
Met een voorbeeld wordt beschreven wat het effect van het onthouden van de niet gekozen situaties is op de route door de zoekboom. De uitleg gebeurt aan de hand van de zoekboom in figuur 5.3.3.1. Hierin zijn twee lagen uitgewerkt waarbij iedere situatie een cijfer heeft gekregen.

Uitwerking:

In het begin is alleen situatie A bekend. Deze wordt dan ook uitgewerkt. Dit levert twee nieuwe situaties op. Dit zijn de situaties B en C. Ze hebben respectievelijk de cijfers zeven en acht. Om het wat korter op te kunnen schrijven wordt dit genoteerd als B(7) en C(8).

Hierna wordt gekeken naar de situatie met het laagste cijfer. Er zijn twee situaties onthouden. Dit B(7) en C(8). Situatie B heeft het laagste cijfer en wordt daarom gekozen. Na uitwerking van situatie B ontstaan de situaties D(9) en E(10). Er zijn nu drie onthouden situatie: C(8), D(9) en E(10). Situatie B(7) is uitgewerkt en daarom uit dit rijtje verwijdert.

Dit is het moment dat er een verschil optreedt in de zoektocht door de zoekboom. Bij het uitvoeren van de Greedy search was de situatie C niet onthouden en werd er verder gezocht met situatie D. Door de situatie te onthouden wordt er nu gekozen verder te zoeken via situatie C omdat deze het laagste cijfer heeft. De uitwerking van deze situatie levert de situaties F en G op. De onthouden situaties zijn nu: D(9), E(10), F(6) en G(8). De volgende situatie die uitgewerkt gaat worden is de situatie F. In figuur 5.3.4.1 is te zien hoe de Best-first search door de zoekboom heen gaat.



Figuur 5.3.4.1: Route door de zoekboom als Best-first search wordt toegepast

Door het onthouden van de situaties wordt voorkomen dat goede vervolgsituaties overgeslagen worden. Dit levert een betere eindoplossing op. Er zit ook een nadeel aan het onthouden van de situaties. Er zijn namelijk meerdere situaties die onthouden worden maar nooit meer gebruikt. Dit is nadelig voor de zoeksnelheid.

5.3.5 Verbeteringen

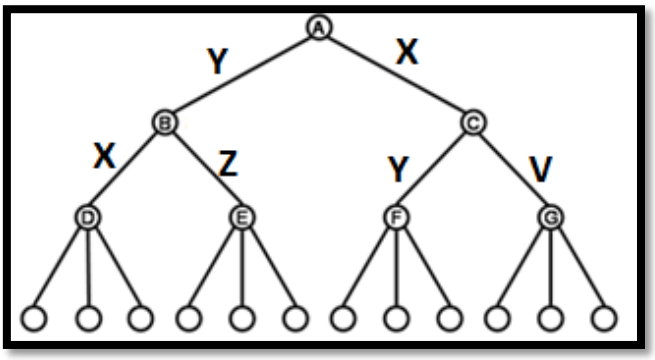
Deze paragraaf beschrijft de verbeteringen op de in de vorige paragrafen behandelde zoekstrategieën. Per verbetering is aangegeven voor welke zoekstrategie het een verbetering is. Naast de inhoud van de verbetering beschrijft de paragraaf ook wat het gevolg van een verbetering is. De verbeteringen zijn:

- Verwijderen van dubbele situaties
- Uitrekenen van meerdere lagen
- Optie beperkingen opstellen

Verwijderen van dubbele situaties:

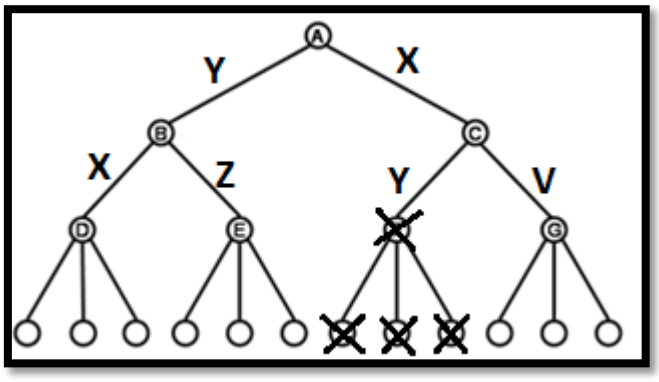
Bij de brute-force search ontstaan na iedere twee verplaatsingen een aantal dubbele situaties. Ze zijn hetzelfde en hebben in het vervolg ook dezelfde opties. Door één van de dubbele situaties te verwijderen verdwijnt de tak eronder ook en wordt onnodig rekenwerk voorkomen. Met een voorbeeld is beschreven wanneer een dubbele situatie kan voorkomen.

Stel dat er een beginsituatie is waarbij situatie A de startsituatie is en de keuze bestaat uit de opties X en Y. Wanneer de keuze valt op de optie Y zijn de volgende opties X en Z. De optie X leidt tot een nieuwe situatie met opties Y en V. De zoekboom van deze situatie is hieronder weergegeven.



Figuur 5.3.5.1: Zoekboom

In de zoekboom zijn er twee situaties identiek aan elkaar. Dit zijn de situaties D en F. Bij beide is er gekozen voor de opties X en Y. Het enige verschil is de volgorde van kiezen. Dit maakt voor de vorm van de situatie niet uit. Een van de twee situaties mag daarom verwijdert worden. In figuur 5.3.5.2 is situatie F weggestreept.



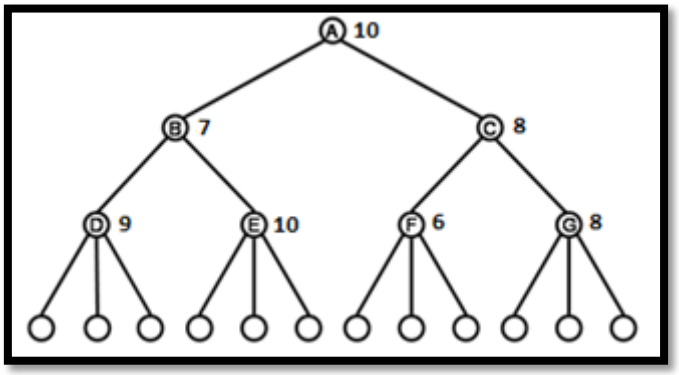
Figuur 5.3.5.2: Zoekboom na verwijdering van situatie F

In het figuur is te zien dat de verwijdering van situatie F ervoor zorgt dat er in die tak geen nieuwe situaties meer aangemaakt worden. Dit scheelt afhankelijk van de grootte van het probleem honderden of zelfs duizenden situaties. Minder situaties aanmaken betekent ook dat er minder situaties onderzocht worden en dat de computer sneller klaar is met doorrekenen.

Deze verbetering is van toepassing op alle vier de zoekstrategieën die beschreven zijn. De theorie is besproken in het filmpje op YouTube waarin Léon Bouquiet het oplossen van een FreeCell-spel uitlegt. Het filmpje heet: 4-3-2015 – Solving any Freecell game using Artificial Intelligence.

Uitrekenen van meerdere lagen:

In paragraaf 5.3.3 is beschreven dat er een nadeel is aan de Greedy search. Het probleem bij de Greedy search is het feit dat de zoekstrategie niet alle situaties aanmaakt en hierdoor de optimale oplossing niet met zekerheid bereikt. In paragraaf 5.3.4 is een zoekstrategie beschreven die dit probleem van de Greedy search oplost. In deze paragraaf wordt een andere aanpassing op de Greedy search beschreven. Deze kan hetzelfde effect hebben als de zoekstrategie uit paragraaf 5.3.4. In figuur 5.3.5.3 is dezelfde zoekboom te zien als in paragraaf 5.3.3.



Figuur 5.3.5.3: Een zoekboom met cijfers

Het probleem is tot op zekere hoogte te verhelpen. Dit kan door niet naar één laag maar naar twee, drie of meer lagen te kijken. Wanneer deze aanpassen tot uitvoering komt op de zoekboom in figuur 5.3.5.3 kijkt de Greedy search naar de eerste twee lagen en maakt de Greedy search de situaties B, C, D, E, F en G aan. Door iedere situatie een cijfer te geven en hierna het laagste cijfer te kiezen gaat de zoektocht naar het optimum verder met situatie F.

Het voordeel van het uitrekenen van meerdere lagen is het vinden van een betere uitgangspositie. Het nadeel is dat er in dit geval eerst twee volledige lagen uitgerekend moeten worden voordat er een keuze gemaakt wordt. Dit kost extra tijd.

Optie beperkingen opstellen:

Het aantal opties dat er per situatie is zorgt voor het aantal nieuwe situaties in de volgende laag. Door het aantal opties te verminderen zakt automatisch ook het aantal situaties. Minder situaties zorgt weer voor een sneller werkend programma.

Bij ieder probleem is het mogelijk regels op te stellen die het aantal opties verkleinen. Omdat het probleem van Anthura Arndt, het spel FreeCell en de schuifpuzzel andere problemen zijn hebben ze ook andere regels. Deze regels zijn specifiek afhankelijk van het probleem en worden later verder uitgewerkt.

5.3.6 Conclusie

Kijkend naar de voor en nadelen van de vier zoekstrategieën is geconcludeerd dat de Best-first search de beste manier is om het probleem van Anthura Arndt op te lossen. Door het juiste cijfer aan de situaties te geven wordt er snel door de zoekboom gezocht naar een oplossing.

Naast het kiezen van de Best-first search als zoekstrategie kunnen er op basis van paragraaf 5.3.5 ook een aantal verbeteringen aan het de zoekstrategie toegevoegd worden. De verbeteringen die het meeste invloed hebben op de snelheid zijn het verwijderen van de dubbele situaties en het beperken van de opties. Het uitrekenen van meerdere lagen is meer een verbetering voor de Greedy Search. Door te kiezen voor de Best-first search is deze verbetering niet van toepassing.

6. Het beginalgoritme

Dit hoofdstuk beschrijft het beginalgoritme dat is opgesteld. Bij het opstellen is rekening gehouden met de uitkomsten van de analyse uit hoofdstuk vier en het literatuuronderzoek uit hoofdstuk vijf. Eerst is beschreven welke zoekstrategie gekozen is. Hierbij is beschreven welke regels er gesteld zijn om het aantal opties te beperken. Hierna is een stukje over de structuur van het geprogrammeerde algoritme beschreven.

6.1 De strategie

Aan de hand van de conclusie over hoofdstuk vijf is gekozen voor een beginalgoritme dat zoekt via de Best-first search. Bij deze zoekstrategie hangt de keuze voor de volgende situatie af van het cijfer dat de situatie gekregen heeft. Dit cijfer wordt ook wel de prioriteit genoemd. In paragraaf 5.3.4 lag een situatie met een laag cijfer dichter bij de oplossing dan een situatie met een hoog cijfer. Dit is onlogisch omdat bijna altijd hoger beter is. Daarom wordt dit in het algoritme omgedraaid. Een situatie met een hoge prioriteit ligt in het algoritme dichter bij de eindsituatie dan een situatie met een lage prioriteit.

De verwachting is dat wanneer alleen de Best-first search ingevoerd wordt het algoritme te veel tijd nodig heeft om een oplossing te genereren. Daarom is besloten om in het beginalgoritme gelijk twee verbeteringen door te voeren. Deze verbeteringen moeten er toe leiden dat het algoritme wel een oplossing genereerd. De verbeteringen die ingevoerd worden:

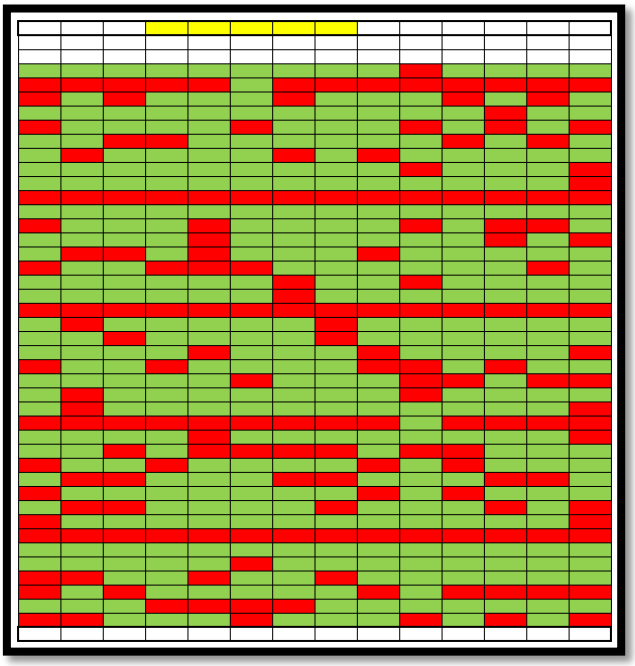
- Het beperken van opties
- Het verwijderen van dubbele situaties

Omdat het de opties en de prioriteit niet voor iedere situatie gelijk zijn wordt er gewerkt in fases. Vanuit een startsituatie moet er door drie fases heen gewerkt worden voordat deze de eindoplossing bereikt. Per fase wordt beschreven wat de opties zijn en hoe de prioriteit opgebouwd is. Allereerst wordt beschreven wat de fases inhouden.

Ook wordt in deze paragraaf beschreven waar de prioriteit op gebaseerd is. De combinatie van factoren die de prioriteit bepaald moet zorgvuldig gekozen worden. Gebeurt dit niet dat wordt er niet efficiënt door de boom heen gezocht. Ook de prioriteiten worden per fase besproken. Het verwijderen van de dubbele situatie staat in paragraaf 5.3.5 beschreven en komt in deze paragraaf niet verder aan bod.

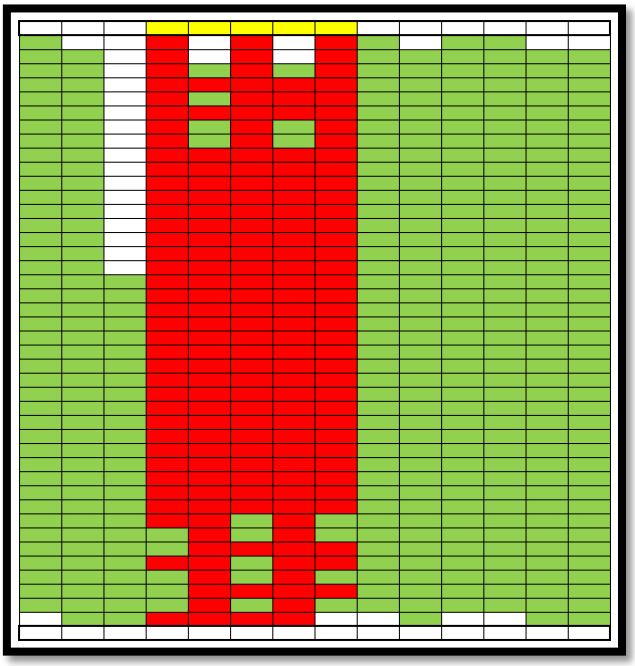
6.1.1 Fases

In de eerste fase zijn de rode rolcontainers verdeeld over alle pijpenbanen. Zowel de voorsorteerbanen als de niet-voorsorteerbanen bevatten rode rolcontainers. In figuur 6.1.1.1 is een voorbeeld van een situatie in deze fase weergegeven.



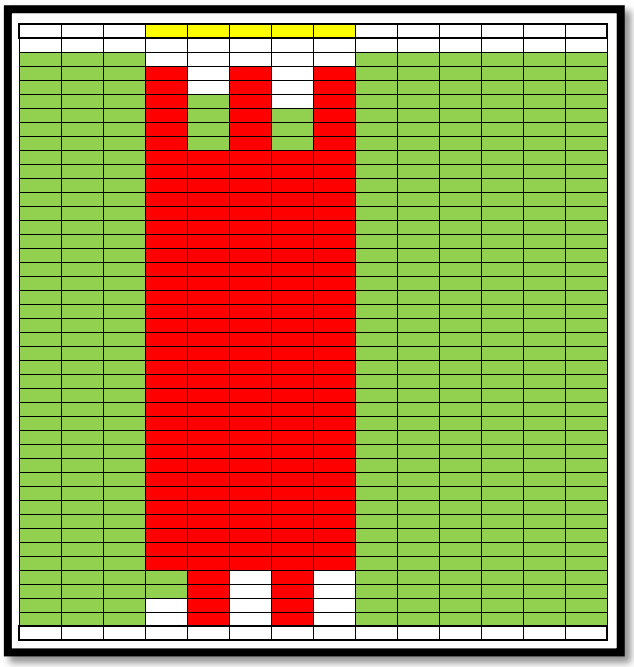
Figuur 6.1.1.1: Voorbeeld van een situatie uit de eerste fase

De tweede fase wordt bereikt door alle rode rolcontainers te verplaatsen naar de voorsorteerbanen. Naast rode rolcontainers staan er in fase 2 ook nog groene rolcontainers op de voorsorteerbanen. Deze groene rolcontainers staan vaak nog tussen de rode rolcontainers in. In figuur 6.1.1.2 is een kasindeling te zien met een situaties die zich in de tweede fase bevindt.



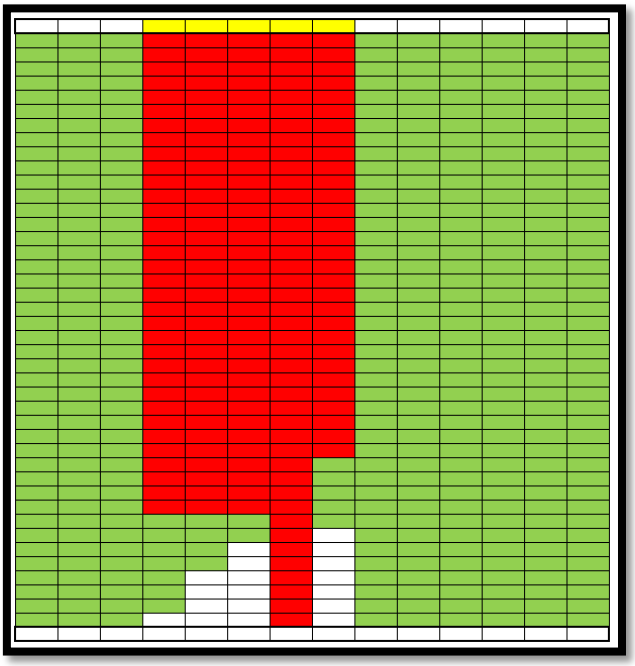
Figuur 6.1.1.2: Voorbeeld van een situatie uit de tweede fase

In de derde fase bevatten zijn de voorsorteerbanen zowel groene als rode rolcontainers. Verschil met fase 2 is dat de groene rolcontainers zich aan één kant van de voorsorteerbaan bevinden. Ze staan nu niet meer tussen de rode rolcontainers in. In figuur 6.1.1.3 is een voorbeeld te zien van een situatie die zich in de derde fase bevindt.



Figuur 6.1.1.3: Voorbeeld van een situatie uit de derde fase

Een situatie bevindt zich in de vierde fase wanneer de eindsituatie bereikt is. Bij de eindsituaties zijn de groene rolcontainers die voor de rode rolcontainers stonden verplaatst en kunnen de rode rolcontainers zonder problemen de voorsorteerbaan verlaten. In figuur 6.1.1.4 is een voorbeeld van een situatie in deze fase te zien.



Figuur 6.1.1.4: Voorbeeld van een situatie uit de vierde fase

6.1.2 Beperken van opties

De beperkingen die aan de opties gesteld zijn worden in deze paragraaf beschreven. Dit gebeurt door eerst de beperkingen te beschrijven die voor alle opties gelden. Hierna worden de beperkingen besproken die voor specifieke fases gelden.

Voor alle fases:
Allereerst wordt er een regel opgesteld omtrent de voorsorteerbanen. Van te voren wordt vastgesteld hoeveel voorsorteerbanen er nodig zijn en in welke richting deze doordraaien. De hoeveelheid voorsorteerbanen hangt af van het aantal rode rolcontainers.

Stel dat er in een situatie tweehonderd rode rolcontainers zijn. De volgende berekening wordt uitgevoerd: $\frac{200}{42} = 4{,}7$. De uitkomst wordt naar boven afgerond en dan is bepaald dat er voor deze startsituatie vijf voorsorteerbanen nodig zijn. In de situaties in de volgende paragraaf zijn ook telkens vijf voorsorteerbanen nodig. Deze zijn aangegeven met een geel vakje aan de bovenkant.

De richting van de voorsorteerbaan wordt aangegeven met een pijl aan de boven- of onderkant van de voorsorteerbaan. De pijl wijst aan waar de rode rolcontainers de voorsorteerbaan in kunnen komen. Aan de andere kant van de voorsorteerbaan verlaten de rolcontainers de baan.

Per fase:
Bij alle vier de fases zijn er andere opties die tot een nieuwe situatie leiden. Deze paragraaf beschrijft wat de toegestane opties in een bepaalde fase zijn. Na het beschrijven van de toegestane opties wordt er een kasindeling laten zien met daarin de namen van de pijpenbanen en de nummers van sommige rolcontainers. In de tabel naast de kasindeling staat welke opties er voor iedere rolcontainer zijn.

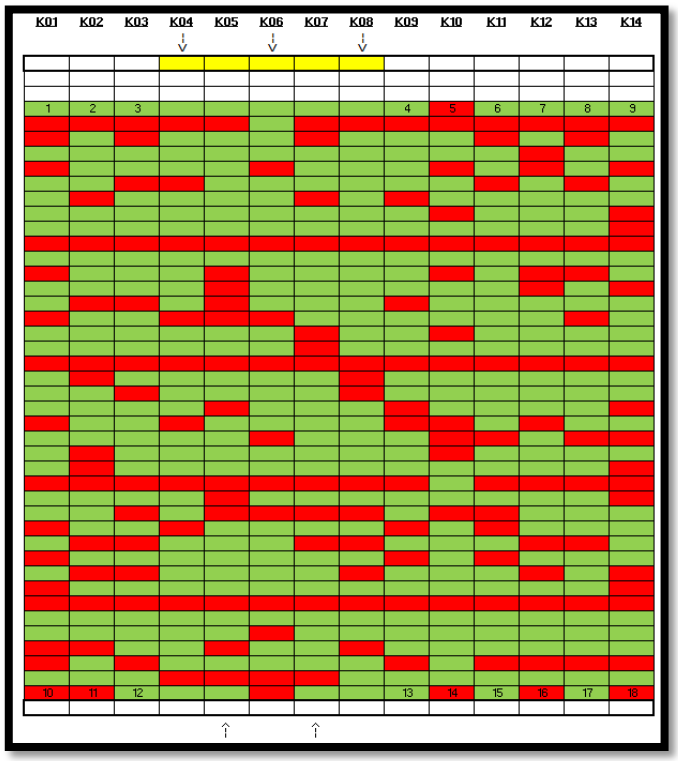
Fase 1:

In de eerste fase is het van belang de rode rolcontainers uit de niet-voorsorteerbanen in de voorsorteerbanen te krijgen. De eerste mogelijke verplaatsing is het verplaatsen van rode rolcontainer uit een niet-voorsorteerbaan naar de voorsorteerbaan. Hierbij worden de rode rolcontainers gelijkmatig over de voorsorteerbanen verdeeld. De rode rolcontainer wordt verplaatst naar de voorsorteerbaan met de minste rode rolcontainers.

Het verplaatsen van een rode rolcontainer kan alleen als er geen groene rolcontainers voor staan. De tweede mogelijke verplaatsing is daarom het verplaatsen van een groene rolcontainer uit een niet-voorsorteerbaan naar een niet-voorsorteerbaan. Hierdoor ontstaat er ruimte voor de rode rolcontainers.

Ook zijn er verplaatsingen uit de voorsorteerbanen mogelijk. Deze zijn echter alleen mogelijk wanneer de voorsorteerbaan vol is. Dit beperkt het aantal lege plaatsen op voorsorteerbanen waardoor er automatisch meer lege plaatsen zijn in een niet-voorsorteerbaan. Dit is belangrijk omdat veel lege plaatsen in niet-voorsorteerbanen resulteert in veel mogelijke verplaatsingen. Er zijn twee verplaatsingen uit een voorsorteerbaan mogelijk. De eerste is een verplaatsing van een rode rolcontainer in de voorsorteerbaan met de minste rode rolcontainers. De andere verplaatsing is de verplaatsing van een groene rolcontainer naar een willekeurige niet-voorsorteerbaan.

In figuur 6.1.2.1 is dezelfde kasindeling te zien als in 6.1.1.1. Naast de kasindeling bevindt zich een tabel met de mogelijk opties.



Rolcontainers	Verplaatsingen mogelijk naar
1	K02, K03, K09, K10, K11, K12, K13, K14
2	K01, K03, K09, K10, K11, K12, K13, K14
3	K01, K02, K09, K10, K11, K12, K13, K14
4	K01, K02, K03, K10, K11, K12, K13, K14
5	K04
6	K01, K02, K03, K09, K10, K12, K13, K14
7	K01, K02, K03, K09, K10, K11, K13, K14
8	K01, K02, K03, K09, K10, K11, K12, K14
9	K01, K02, K03, K09, K10, K11, K12, K13
10	K05
11	K05
12	K01, K02, K09, K10, K11, K12, K13, K14
13	K01, K02, K03, K10, K11, K12, K13, K14
14	K05
15	K01, K02, K03, K09, K10, K12, K13, K14
16	K05
17	K01, K02, K03, K09, K10, K11, K12, K14
18	K05

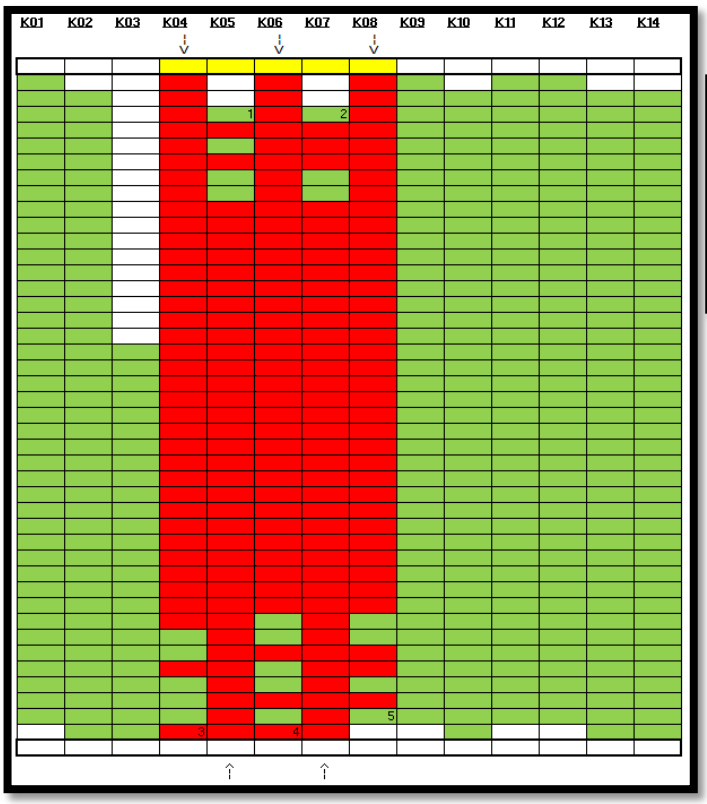
Figuur 6.1.2.1: Links een kasindeling, rechts de opties die horen bij de kasindeling

Fase 2:

In de tweede fase moeten de rolcontainers op zo'n manier verplaatst worden dat er op de voorsorteerbanen geen groene rolcontainers meer tussen de rode rolcontainers in staan. Om dit te bereiken zijn drie soorten verplaatsingen mogelijk. De eerste is het verplaatsen van een rode rolcontainer uit een voorsorteerbaan in de voorsorteerbaan met de minste rode rolcontainers. Hiervoor hoeft de huidige voorsorteerbaan niet meer vol te zijn.

De tweede verplaatsing is het verplaatsen van een groene rolcontainer vanuit een voorsorteerbaan naar een niet-voorsorteerbaan. De enige criteria hiervoor is het hebben van een niet-voorsorteerbaan met een lege plaats. Is deze er niet dan kan de derde verplaatsing uitgevoerd worden. Dit is een verplaatsing waarbij een groene rolcontainer vanuit een niet-voorsorteerbaan verplaatst wordt naar een voorsorteerbaan. Deze verplaatsing is er zodat er ruimte gemaakt wordt voor andere groene rolcontainers in de niet-voorsorteerbanen. Van te voren wordt één voorsorteerbaan gekozen die als extra opslag voor groene rolcontainers geldt. Dit is een van de voorsorteerbanen waar de pijl boven de voorsorteerbaan staat. De rolcontainer wordt via de onderkant de voorsorteerbaan ingebracht.

In figuur 6.1.2.2 is dezelfde kasindeling te zien als in figuur 6.1.1.2. In de tabel ernaast is wederom weergegeven wat de opties zijn in deze situatie.



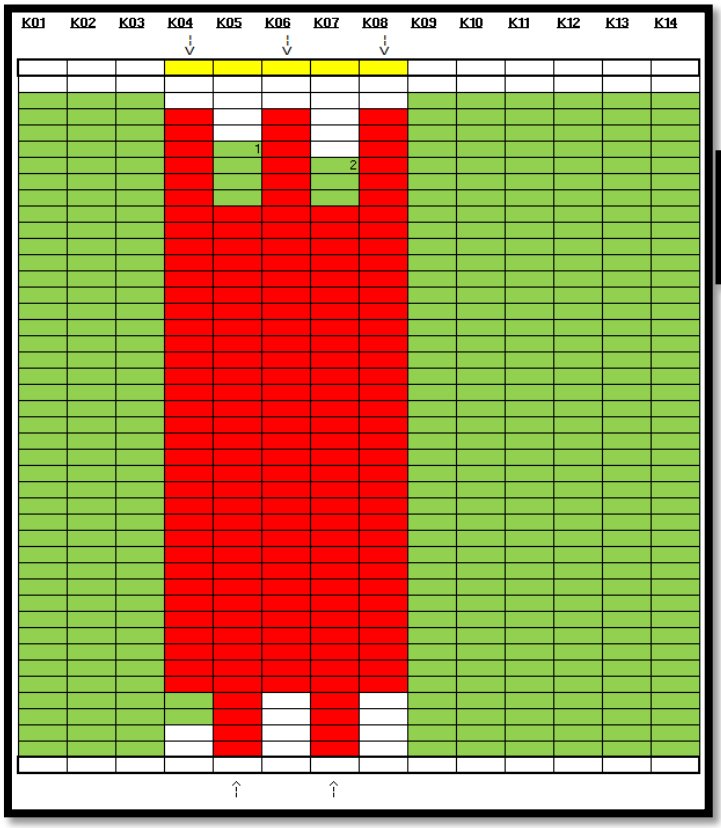
Rolcontainers	Verplaatsingen mogelijk naar
1	K01, K02, K03, K09, K10, K11, K12, K13 en K14
2	K01, K02, K03, K09, K10, K11, K12, K13 en K14
3	K05
4	K05
5	K01, K02, K03, K09, K10, K11, K12, K13 en K14

Figuur 6.1.2.2: Links een kasindeling, rechts de opties die horen bij de kasindeling

Fase 3:

In de derde fase moeten alleen de groene rolcontainers die voor de rode rolcontainers staan verplaatst worden. Er zijn twee verplaatsingen die dit mogelijk maken. Allereerst kan een groene rolcontainer vanuit een voorsorteerbaan naar een willekeurige niet-voorsorteerbaan verplaatst worden. Dit kan natuurlijk alleen als er ruimte op de niet-voorsorteerbanen is. Is dit er niet dan wordt deze gemaakt door een groene rolcontainer vanuit een niet-voorsorteerbaan naar de onderkant van een voorsorteerbaan te verplaatsen. Deze voorsorteerbaan wordt van te voren bepaald.

In figuur 6.1.2.3 is dezelfde kasindeling te zien als in figuur 6.1.1.3. In de tabel ernaast is weergegeven wat de opties in deze situatie zijn.

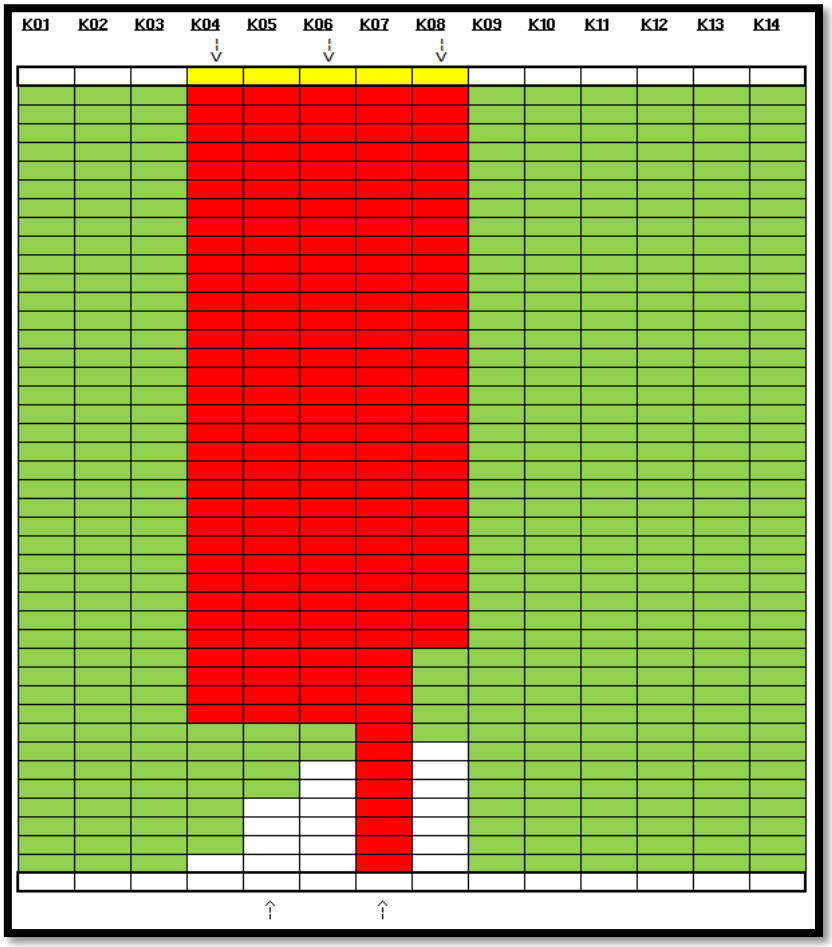


Rolcontainers	Verplaatsingen mogelijk naar
1	K01, K02, K03, K09, K10, K11, K12, K13, K14
2	K01, K02, K03, K09, K10, K11, K12, K13, K14

Figuur 6.1.2.3: Links een kasindeling, rechts de opties die horen bij de kasindeling

Fase 4:

Bevindt een situatie zich in de vierde fase dan is de eindsituatie bereikt. De rode rolcontainers staan op de voorsorteerbanen en er staat geen groene rolcontainers meer in de weg. In deze situatie zijn er dan ook geen verplaatsingen meer mogelijk. In figuur 6.1.2.4 is een kasindeling te zien die in de vierde fase zit.



Figuur 6.1.2.4: Een kasindeling behorend bij de vierde fase

6.1.3 Opstellen cijfer

Het cijfer dat de prioriteit van een situatie bepaald is opgebouwd uit meerdere factoren. Net als bij het beperken van de opties heeft iedere fase andere factoren die van invloed zijn op het cijfer. Voor sommige factoren geldt dat ze in alle fases van invloed zijn. Daarom wordt niet per fase beschreven welke factoren hierbij van invloed zijn maar worden de factoren één voor één uitgeschreven. Hierna worden de factoreen aan een fase toegekend. Er komen in deze paragraaf drie factoren aan bod:

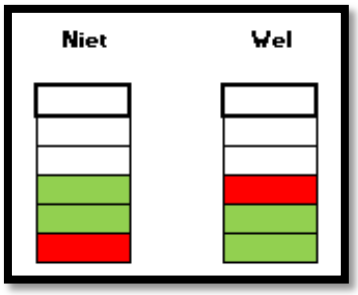
- Rode rolcontainer in een voorsorteerbaan
- Positie van de voorste rode rolcontainer
- Lege plaats op een pijpenbaan

Rode rolcontainer in voorsorteerbaan:

Het doel van het algoritme is het verplaatsen van de rode rolcontainers naar de voorsorteerbanen. Het aantal rode rolcontainers in de voorsorteerbanen geeft aan hoe dicht het algoritme bij de eindoplossing is. Omdat dit de belangrijkste verplaatsing is krijgt een rode rolcontainer die in een voorsorteerbaan staat veel punten. Het aantal punten is vastgesteld op +400. Dit zodat deze factor altijd het meeste effect heeft op de keuze voor de volgende situatie. Een situatie waarbij een rode rolcontainer vanuit een niet-voorsorteerbaan naar een voorsorteerbaan verplaatst is heeft een hoger cijfer dan de voorgaande situatie en is daarom dichter bij de eindsituatie.

Positie van de voorste rode rolcontainer:

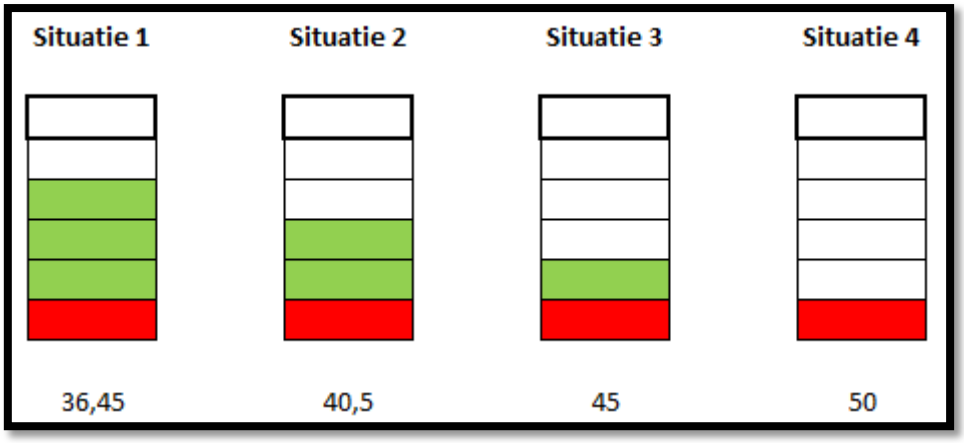
Het kan zijn dat er geen opties zijn waarbij rode rolcontainers naar een voorsorteerbaan verplaatst kan worden. Dit komt doordat er groene rolcontainers zijn die voor de rode rolcontainers staan. Deze groene rolcontainers moeten dan eerst verplaatst worden voordat de rode rolcontainer verplaatst kan worden. In figuur 6.1.3.1 is een voorbeeld te zien van een rode rolcontainer die niet verplaatst kan worden (links) en een rode rolcontainer die wel verplaatst kan worden (rechts).



Figuur 6.1.3.1: Niet te verplaatsen rode rolcontainer links. Recht een wel te verplaatsen rode rolcontainer.

Door punten te geven voor het verplaatsen van deze groene rolcontainers wordt vanzelf de rechter situatie in figuur 6.1.3.1 gecreëerd. Dit gebeurt door punten toe te kennen aan de positie die de eerste rode rolcontainer in de pijpenbaan heeft ten opzichte van de laatste lege plek op de pijpenbaan. Met een voorbeeld wordt dit duidelijk.

In figuur 6.1.3.2 zijn vier mogelijk posities van een rode rolcontainer op een pijpenbaan te zien. Bij de eerste bevindt de rode rolcontainer zich op de vierde positie op de pijpenbaan. Bij de andere bevindt de rode rolcontainer zich respectievelijk op de derde, tweede en eerste positie op de pijpenbaan. Onder de situaties zijn de punten voor de positie weergegeven.



6.1.3.2: De rode rolcontainers op verschillende posities met de punten

Is de eerste situatie zijn er 36,45 punten gegeven voor een rode rolcontainer op de vierde positie. In de tweede situatie zijn er 40,5 punten gegeven voor een rode rolcontainer op de derde positie. Dit betekent dat de verplaatsing van een groene rolcontainer die situatie één veranderd in situatie twee $40{,}5 - 36{,}45 = 4{,}05\ punten$ oplevert. Een verplaatsing die situatie twee in situatie drie veranderd levert 4,5 punten op en een verplaatsing die situatie drie in situatie vier veranderd lever 5 punten op.

Een verplaatsing die een rode rolcontainer op positie twee zet krijgt meer punten dan een verplaatsing die de rode rolcontainer op positie vier zet. Door de punten zo te verdelen krijgt de verplaatsing die zorgt voor de laagste positie de meeste voorkeur. Hierdoor wordt voor de eerste verplaatsing gekozen en komt vanzelf de rode rolcontainer op de eerste positie te staan. De formule voor het aantal punten is bepaald door veel te testen met steeds verschillende waardes. De formule is:

$$Punten = 50 * 0.9^{de\ positie\ van\ de\ rode\ rolcontainer - 1}$$

Lege plaats op pijpenbaan:

Het is belangrijk dat er per situatie zoveel mogelijk toegestane opties zijn. Een volle pijpenbaan beperkt het aantal toegestane opties. Door een volle pijpenbaan kan het bijvoorbeeld dat een verplaatsing van een rode rolcontainer naar een voorsorteerbaan niet mogelijk is omdat deze vol is. Door punten te geven voor een pijpenbaan met een lege plaats zorgt de best-first search er automatisch voor dat veel pijpenbanen een lege plaats hebben.

Het geven van punten is verschillend voor een voorsorteerbaan en een niet-voorsorteerbaan. Dit omdat het belang van een lege plaats op een voorsorteerbaan groter is dan een lege plaats op een niet-voorsorteerbaan. Ook hier zijn de formules voor de punten bepaald door veel testen met verschillende waardes in de formule. De formules zijn:

$$Punten\ voorsorteerbaan = Aantal\ voorsorteerbanen\ met\ een\ lege\ plaats * 100$$

$$Punten\ nietvoorsorteerbaan = Aantal\ nietvoorsorteerbanen\ met\ een\ lege\ plaats * 40$$

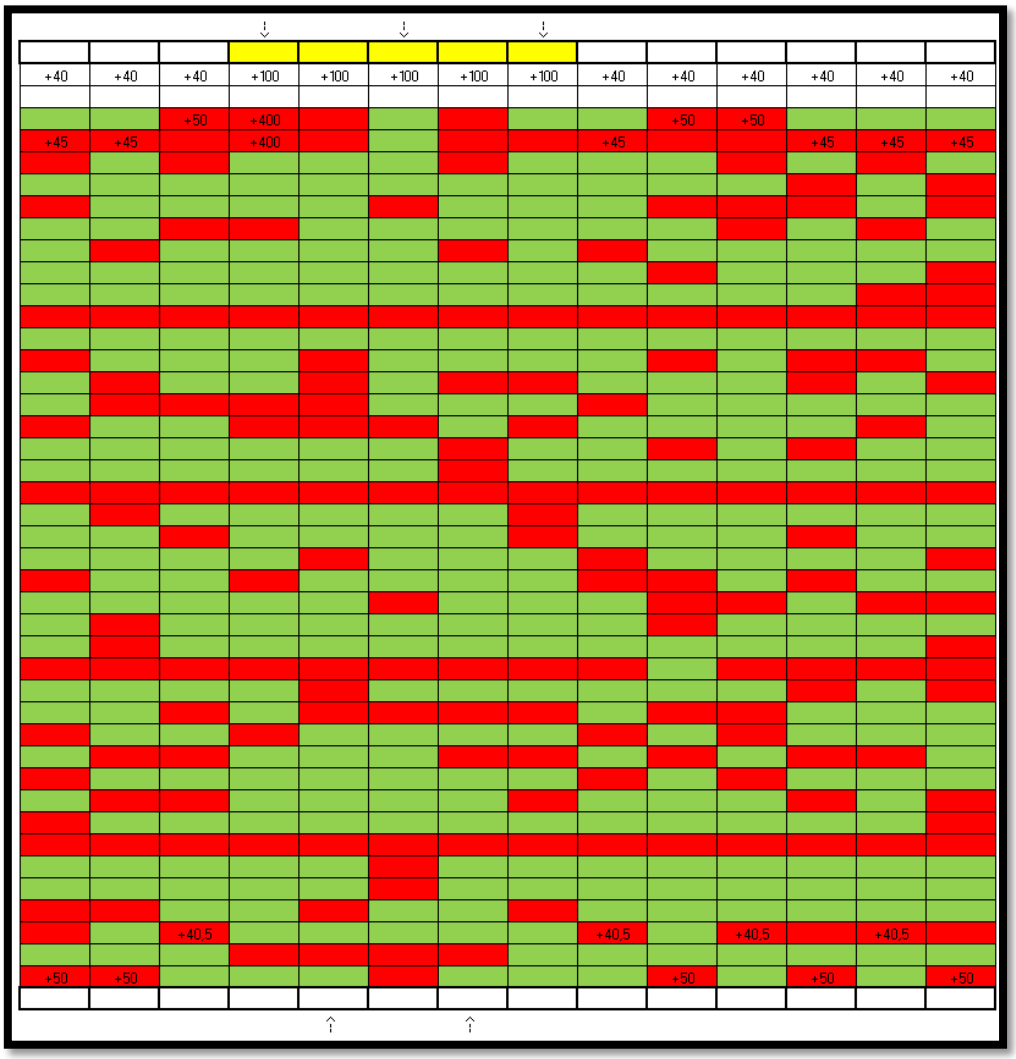
Alle factoren zijn beschreven. Hieronder is per fase aangegeven welke factoren er meegenomen worden in het prioriteitencijfer. Daarnaast is bij iedere fase een figuur ingevoegd met een kasindeling die in deze fase hoort. De prioriteit punten zijn in de kasindeling weergegeven.

Fase 1:

Tussen de eerste en de tweede fase moeten de rode rolcontainers naar de voorsorteerbanen verplaatst worden. Iedere rode rolcontainer in een voorsorteerbaan waarbij een pijl naar de rolcontainer wijst levert daarom punten op. Daarnaast is het in deze fase belangrijk dat de rode rolcontainer in de niet-voorsorteerbanen vooraan komen te staan. Daarom worden er ook punten gegeven voor de positie die een rode rolcontainer in een niet-voorsorteerbaan heeft. Als laatste is het van belang dat er geen beperkingen zijn die zorgen voor minder opties. Een pijpenbaan met een lege plek krijgt daarom ook punten, De punten opgesomd:

- Rode rolcontainer naar voorsorteerbaan (+400)
- Positie van de voorste rode rolcontainer in een niet-voorsorteerbaan (max +50)
- Lege plaats op voorsorteerbaan (+100)
- Lege plaats op niet-voorsorteerbaan (+40)

In figuur 6.1.3.3 is een kasindeling te zien die hoort bij de eerste fase. Hierin is het aantal punten weergegeven. Opgeteld komt dit neer op een prioriteit cijfer van +2.492.



Figuur 6.1.3.3: Kasindeling uit fase 1. Het aantal punten is in de vakjes weergegeven

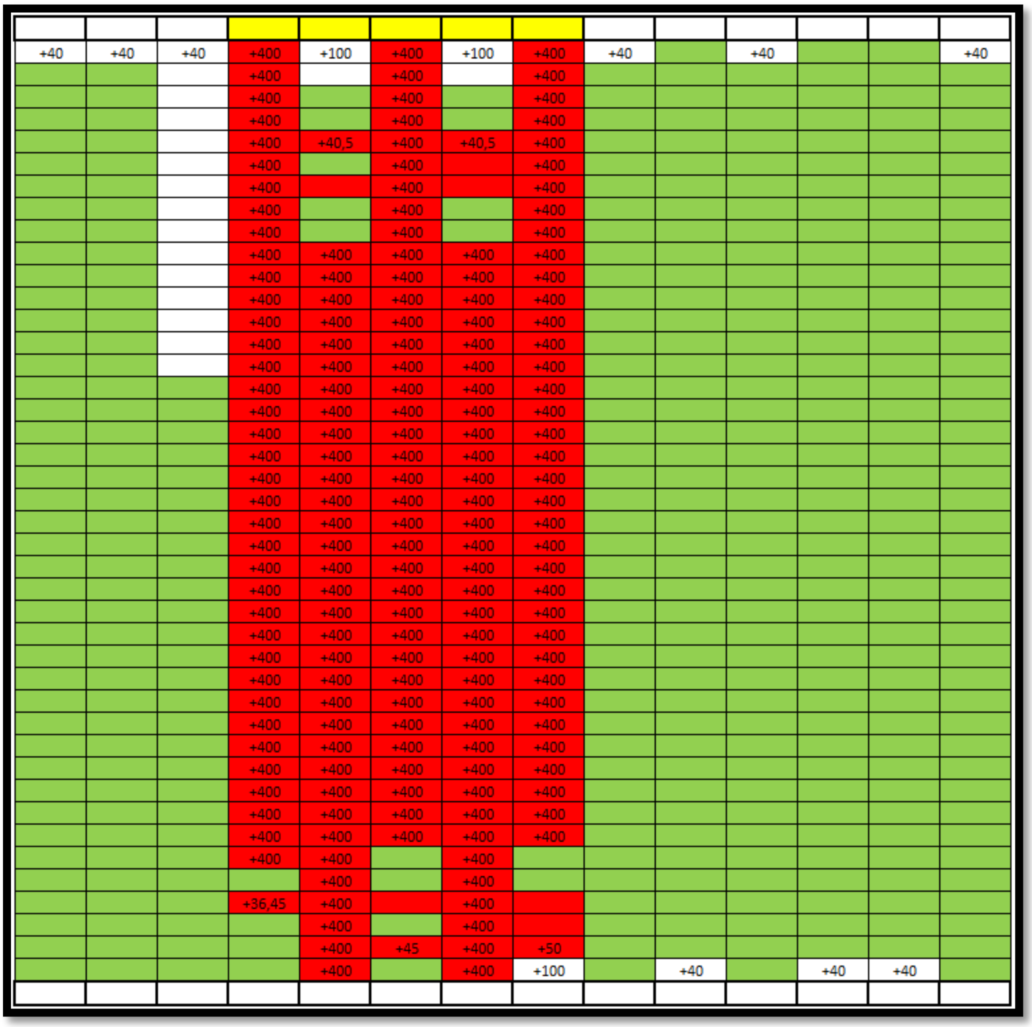
Fase 2:

In de tweede fase staan er geen rode rolcontainers in de niet-voorsorteerbanen. Punten voor de positie van een rode rolcontainer in een niet-voorsorteerbaan heeft daarom geen nut. Er staan immers toch geen rode rolcontainers meer in de niet-voorsorteerbanen.

Wel worden er nog steeds punten gegeven voor iedere rode rolcontainer die in een voorsorteerbaan staat. Daarnaast worden er in de voorsorteerbanen punten gegeven voor de posities die de voorste rode rolcontainer heeft. Ook is het nog steeds van belang dat er lege plaatsen op de pijpenbanen aanwezig zijn. De factoren opgesomd:

- Rode rolcontainer naar voorsorteerbaan (+400)
- Positie van de voorste rode rolcontainer in een voorsorteerbaan (max +50)
- Lege plaats op voorsorteerbaan (+100)
- Lege plaats op niet-voorsorteerbaan (+40)

In figuur 6.1.3.4 is een kasindeling te zien die hoort bij de eerste fase. Opgeteld heeft deze situatie het cijfer +70872,45.



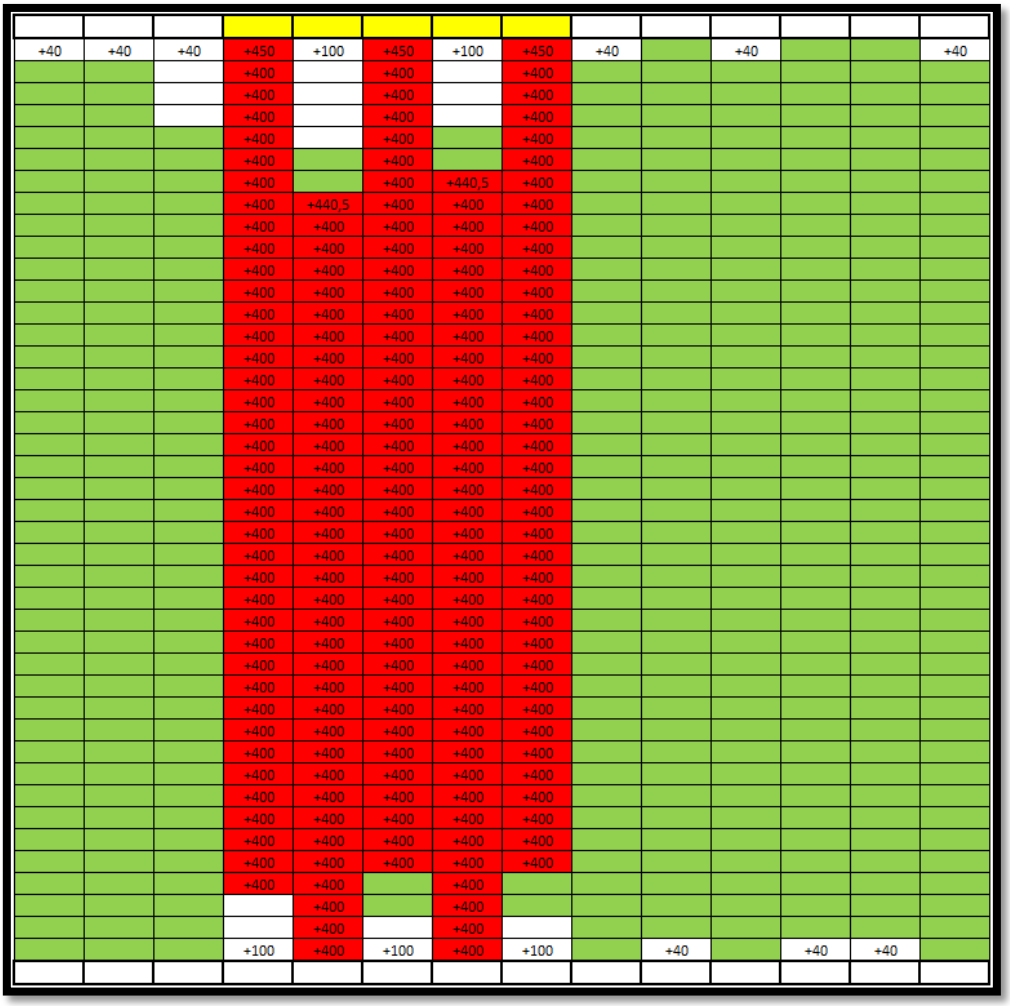
Figuur 6.1.3.4: Kasindeling uit fase 2. Het aantal punten is in de vakjes weergegeven

Fase 3:

In de derde fase moeten alleen de groene rolcontainers aan de voorkant verplaatst worden. Nog steeds worden er punten gegeven voor een rode rolcontainer in een voorsorteerbaan. Daarnaast krijgen de rode rolcontainers aan de voorkant van de voorsorteerbaan punten voor hun positie. Dit geldt alleen aan de voorkant van de voorsorteerbaan omdat hier geen groene rolcontainers meer voor de rode rolcontainers mogen staan. Aan de achterkant maakt dit niet uit. De factoren opgesomd:

- Positie van de voorste rode rolcontainer aan de voorkant in een voorsorteerbaan (max +50)
- Rode rolcontainer naar voorsorteerbaan (+400)
- Lege plaats op voorsorteerbaan (+100)
- Lege plaats op niet-voorsorteerbaan (+40)

In figuur 6.1.3.5 is een situatie die hoort bij de derde fase uitgewerkt. Het cijfer voor deze situatie is +75491.



Figuur 6.1.3.5: Kasindeling uit fase 3. Het aantal punten is in de vakjes weergegeven

Fase 4:

In fase vier is de eindsituatie bereikt. Daarom zijn er geen verplaatsingen mogelijk en is het geven van punten zinloos.

6.2 De structuur van het programma

Tijdens het programmeren van het algoritme zijn er vele keuzes gemaakt. In deze paragraaf zijn de belangrijkste keuzes beschreven. De is zijn:

- Gebruik van een lijst voor het op volgorde opslaan van situaties
- Gebruik van een Dictionary voor het filteren van dubbele situaties
- Gebruik van een Omzetter voor de dataopslag en de vergelijking

6.2.1 Lijst

De eerste keuze die gemaakt is betreft het opslaan van de situaties. Na het uitwerken van een situatie moeten de nieuwe situaties ergens in opgeslagen worden. Ze worden onthouden door het programma. In het filmpje van Leon Bouquiet werden de spelsituaties opgeslagen in een Queue. Een Queue is een wachtrij in C# die werkt volgens het principe first in, first out. De voorste situatie in de wachtrij werd uitgewerkt en de situaties die hieruit ontstonden werden achteraan de wachtrij ingevoegd. Aanvankelijk was hierom gekozen om bij het opslaan een Queue te gebruikten.

Door het gebruik van de Best-first search was de Queue niet meer handig om te gebruiken. Bij de Best-first search moet de wachtrij namelijk op prioriteit ingedeeld zijn. De situatie met de hoogste prioriteit vooraan en de situatie met de laagste prioriteit achteraan. De Queue zet de laatst binnengekomen situatie achteraan zonder naar de prioriteit gekozen.

Om deze reden is er gekozen voor het gebruik van een Lijst om de situaties in op te slaan. Een Lijst is een rij met daarin allemaal objecten van één bepaalde variabele. In dit geval is de variabele de kasindeling van de dertiende afdeling. De objecten zijn allemaal specifieke situaties van deze dertiende afdeling. Iedere keer als er een situatie aangemaakt is word er naar de prioriteit gekeken en wordt deze op de juiste plaats ingevoerd in de Lijst. Iedere uitgewerkte situatie wordt uit de Lijst verwijdert.

6.2.2 Dictionary

Voor het vergelijken van variabelen heeft C# een eigen object. Dit object heet een Dictionary. In de Dictionary worden alle situaties opgeslagen. Er ontstaat een rij met situaties. Dit lijkt veel op de Lijst zoals beschreven in paragraaf 6.2.1. Er is echter een belangrijk verschil tussen een Lijst en een Dictionary. De Dictionary heeft namelijk een eigen functie waarmee bekeken kan worden of een bepaalde situatie al in de Dictionary aanwezig is. Door deze functie aan te roepen en de situatie mee te geven wordt bepaalt of de situatie al in de Dictionary aanwezig is of niet. Een ander verschil is het feit dat alle situaties in de Dictionary blijven. Ook de eerder uitgewerkte situaties blijven aanwezig. Hierdoor kan makkelijk onderzocht worden of een situatie al eerder aangemaakt is en of het dus dubbele situatie is.

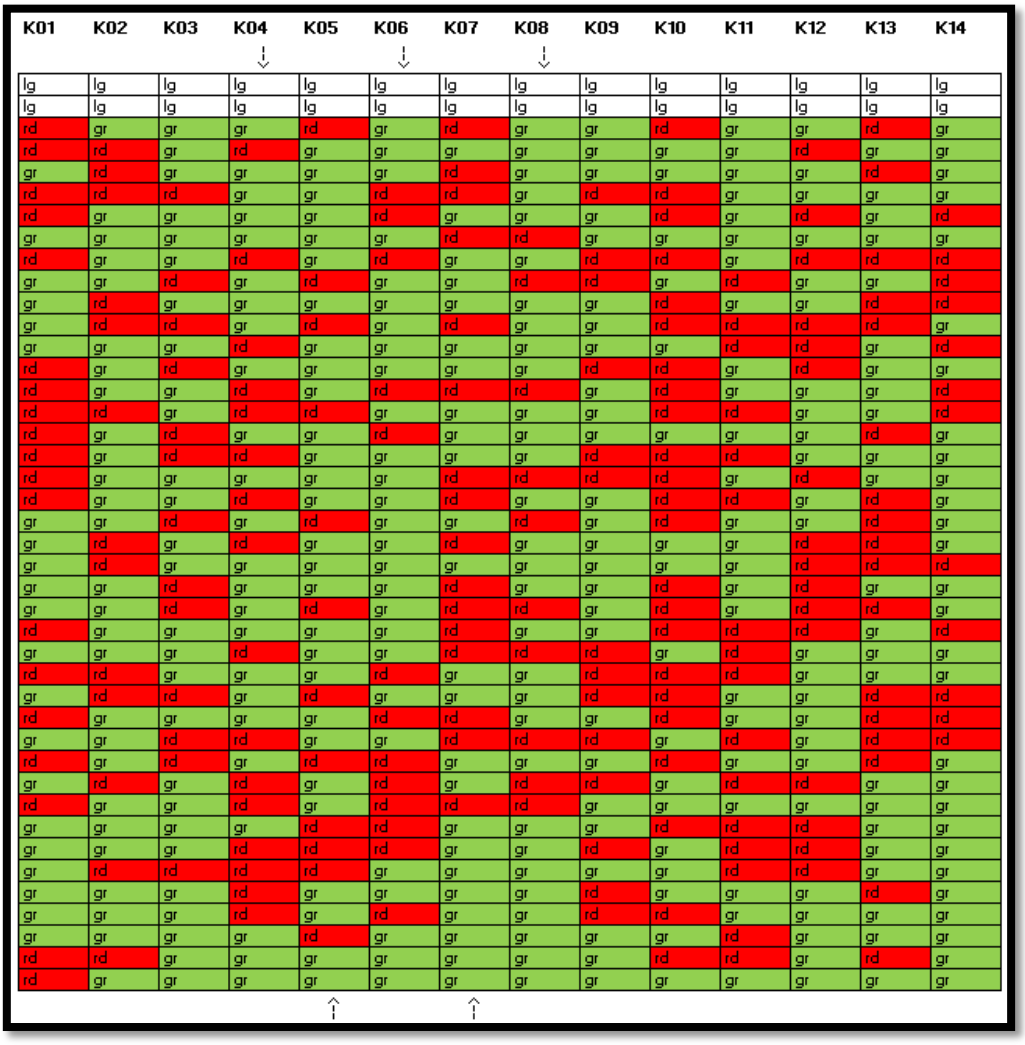
6.2.3 Omzetter

Wanneer de situatie als een stuk tekst ingevoerd wordt kan de Dictionary deze omzetten in een code. Deze code is uniek en kan daardoor als vergelijkingsmateriaal gebruikt worden. Echter, de situatie die aan de Dictionary toegevoegd moet worden is niet opgeslagen als een tekst. Het omschrijven van een situatie naar code kan de Dictionary hierdoor niet. Door een functie te maken waarin de situatie omgeschreven wordt naar een stuk tekst en deze in te voeren in de Dictionary kan deze hem wel vergelijken. Hieronder is beschreven hoe het omzetten precies werkt.

Een situatie bestaat uit veertien aparte pijpenbanen met op iedere baan 42 locaties. In het programma heten deze locaties:

- “lg” als er geen rolcontainer op deze locatie aanwezig is
- “gr” als er een rolcontainer zonder orders op de locatie aanwezig is
- “rd” als er een rolcontainer met orders op de locatie aanwezig is

Door de “lg”, “gr” en “rd” om te zetten naar respectievelijk “0”, “1” en “2” en deze hierna achter elkaar te plakken ontstaat er een tekst van allemaal cijfers. Deze tekst kan de Dictionary wel omzetten naar code. In figuur 6.2.3.1 is een voorbeeldsituatie te zien met in de vakjes de namen van de locaties.



Figuur 6.2.3.1: Situatie met “lg”, “gr”, “rd”

De tekst met alle cijfer is hieronder per pad weergegeven:

- K01: 00221221211112222222111112121212121111122
- K02: 00122211122111211111221111221112111211121
- K03: 00111211121212112211211221112122111121111
- K04: 00121111211121221212121111211121221222211
- K05: 00211111121211121111211121112112112221121
- K06: 00111221211111212111111111121212222211211
- K07: 00212212111211211221212222112211211111111
- K08: 00111112121111211121211121211121221111111
- K09: 00111211221112111221111111222121211212211
- K10: 00211221212212221222211222122212112111212
- K11: 00111111121221121212111112221121212221122
- K12: 00121121211222111121122222111111212221111
- K13: 00212111212211112112222121112222211111212
- K14: 00111121222121221111112112112221111111111

Wanneer van alle veertien paden de teksten achter elkaar geplakt worden ontstaat de tekst die in de Dictionary ingevoerd wordt.

6.3 Het testrapport

Na het programmeren van het beginalgoritme is deze getest op drie onderdelen. Dit zijn de snelheid van het algoritme, de efficiëntieafwijking van de oplossing en de uitvoerbaarheid van de oplossing. Welke testen er gedaan zijn en de resultaten van deze testen staan in dit hoofdstuk beschreven.

6.3.1 De tests

Het testen is gedaan door het algoritme meerdere keren uit te voeren. Telkens werd er met een andere startsituatie begonnen. Na een korte analyse is het aantal startsituaties bepaald op negentien. In deze paragraaf is een startsituatie gelijk aan een test. Beschreven wordt met welke aantallen rode rolcontainers er getest wordt, hoeveel situaties er per aantal rode rolcontainers getest zijn, hoe vaak een test uitgevoerd is en hoelang een test duurt.

Aantallen rolcontainers:

Allereerst wordt beschreven met welk aantal rolcontainers er getest is. Iedere startsituatie heeft een ander aantal rode rolcontainers. Hieronder is beschreven om welke aantallen het gaat en waarom er voor deze aantallen gekozen is.

Test 1 (294 rode rolcontainers)

Allereerst is er getest of het algoritme een oplossing vindt wanneer alle voorsorteerbanen gebruikt worden met een maximale capaciteit. In paragraaf 4.1.1 is beschreven dat er in totaal zeven voorsorteerbanen zijn en dat er op iedere pijpenbaan ruimte is voor 42 rolcontainers. Dit betekent dat er in deze test 42*7=294 rolcontainers met orders in de kas aanwezig zijn.

Test 2 (211 rode rolcontainers) en Test 3 (169 rode rolcontainers)

Bij de tweede en derde test is het aantal rode rolcontainers zo gekozen dat één rode rolcontainer minder zorgt voor één voorsorteerbaan minder. Wat hiermee bedoeld wordt is dat er bij 211 en 169 rode rolcontainers een voorsorteerbaan meer nodig is dan bij 210 en 168 rode rolcontainers. Dit betekent dat er bij beide testen 41 plekken op de voorsorteerbanen zijn waar geen rode rolcontainers komen te staan.

Nu klinkt het alsof de twee testen hetzelfde zijn. Dit is niet het geval. Bij test 2 wordt getest of de code werkt wanneer er 41 lege plekken zijn op een even aantal voorsorteerbanen zijn. Bij test 3 wordt getest of de code werkt wanneer er 41 lege plekken zijn op een oneven aantal voorsorteerbanen zijn. Door even en oneven te test worden alle situaties opgevangen.

Test 4 t/m Test 19

Vooraf heeft Logiqs B.V. aangegeven dat het aantal rode rolcontainers tussen de 100 en 250 ligt. Dit betekent dat de code moet werken voor alle aantallen rolcontainers tussen de 100 en 250. Omdat het te veel werk is om alle aantallen te testen is ervoor gekozen de aantallen af te ronden op tientallen en deze te testen. Het resultaat zijn de volgende tests:

- Test 4 (100 rode rolcontainers)
- Test 5 (110 rode rolcontainers)
- Test 6 (120 rode rolcontainers)
- Test 7 (130 rode rolcontainers)
- Test 8 (140 rode rolcontainers)
- Test 9 (150 rode rolcontainers)
- Test 10 (160 rode rolcontainers)
- Test 11 (170 rode rolcontainers)
- Test 12 (180 rode rolcontainers)
- Test 13 (190 rode rolcontainers)
- Test 14 (200 rode rolcontainers)
- Test 15 (210 rode rolcontainers)
- Test 16 (220 rode rolcontainers)
- Test 17 (230 rode rolcontainers)
- Test 18 (240 rode rolcontainers)
- Test 19 (250 rode rolcontainers)

Aantallen situaties:

Er zijn voor ieder aantal rode rolcontainers meerdere mogelijkheden om deze over de kas te verdelen. Er zijn dus ook meerdere startsituaties per test mogelijk. De resultaten uit één startsituatie zijn daarom niet representatief voor de hele populatie van startsituaties van de betreffende test. Het testen van alle startsituaties is simpelweg te veel. Voor een test met honderd rode rolcontainers zijn bijvoorbeeld $5.75 * 10^{112}$ startsituaties mogelijk. Een test met 250 rode rolcontainers heeft $5.10 * 10^{165}$ mogelijke startsituaties. Omdat het uitvoeren van een test veel tijd kost is ervoor gekozen om bij iedere test drie startsituaties te maken. Bij meer dan drie startsituaties per test neemt het testen te veel tijd in beslag.

Tijdsduur van de tests:

Het is mogelijk dat een test door externe factoren langer duurt dan gemiddeld. Een goede conclusie over de tijdsduur van het algoritme kan dan eigenlijk niet gemaakt worden. Om deze reden worden alle startsituaties drie maal uitgevoerd. De conclusie over de uitvoerbaarheid en efficiëntieafwijking blijven hetzelfde. De snelheid verschilt wel. Om een goede conclusie over de tijdsduur te kunnen maken wordt de gemiddelde snelheid van de drie testen genomen. Als laatste moet de test ergens een keer stoppen. Omdat het programma snel een uitkomst moet hebben is het niet nodig het programma lang te laten draaien. Daarom is vastgesteld dat iedere test na drie minuten stopt.

Samenvattend:

De duur van het uitvoeren van alle tests is daarmee
$Aantal\ tests(19) * Aantal\ situaties(3) * Aantal\ maal\ uitvoeren(3) * Duur\ test(3) = 513\ minuten.$

6.3.2 De resultaten

De resultaten van de tests zijn in bijlage I weergegeven. Op de resultaten zijn er drie analyses gedaan. In deze paragraaf zijn deze analyse beschreven. De analyses die gedaan zijn:

- Uitvoerbaarheidsanalyse
- Tijdsanalyse
- Efficiëntie-analyse

Bij de uitvoerbaarheidsanalyse wordt bekeken of de tests een oplossing hebben. Een oplossing bestaat uit de verplaatsingen die nodig zijn om van de startsituatie naar de eindsituatie te aan. Heeft de test een oplossing dan wordt er bepaald wat de uitkomst is als de verplaatsingen uitgevoerd worden op de startsituatie. Is de situatie die hieruit ontstaat de eindsituatie dan is de oplossing van de test uitvoerbaar. Heeft de test geen oplossing dan moet worden uitgezocht waar het programma op vast loopt.

Van te voren is er een streeftijd vastgelegd. Bij de tijdsanalyse wordt bekeken of het programma aan deze streeftijd voldoet. Hierbij is het belangrijk dat de uitkomsten in perspectief gebracht worden. De rekenkracht van de computer heeft namelijk heel veel invloed op de snelheid van het programma. Bij het testen is er gebruik gemaakt van een laptop. De vastgestelde streeftijd geldt dan ook voor deze laptop. De computers van Logiqs B.V. hebben een grotere rekenkracht. Dit betekent dat het uitvoeren van het programma op een computer van Logiqs B.V. automatisch resulteert in een sneller programma.

Als laatste wordt er een efficiëntie-analyse gedaan. Hierbij wordt per test bekeken of het aantal verplaatsingen dat de gekregen oplossing heeft in de buurt ligt van het minimum. Ligt de oplossing hier dichtbij dan is het een goede oplossing. Omdat het minimum heel erg moeilijk te bepalen is wordt er bij deze analyse voor iedere situatie een ondergrens voor het aantal verplaatsingen bepaald. Over deze grens kan gezegd worden dat het minimum aantal verplaatsingen wel boven maar niet onder deze grens ligt. Aan de hand van de ondergrens wordt bepaald of de oplossing efficiënt is.

Uitvoerbaarheidsanalyse:

Bij de uitvoerbaarheidsanalyse wordt allereerst bekeken of de tests een oplossing genereren. Van de 57 startsituaties zijn er drie waarbij het algoritme niet binnen drie minuten een oplossing gegenereerd heeft. Voor alle andere startsituaties zijn er wel oplossingen. Voor deze oplossingen moet uitgezocht worden of de verplaatsingen die uit de test komen tot de juiste eindsituatie leiden. Dit wordt getest met een programma in Excel. In dit programma wordt de output van de test ingevoerd. De output is:

- Een overzicht van de beginsituatie
- Een overzicht van de eindsituatie
- De verplaatsingen die nodig zijn om van beginsituatie in de eindsituatie te komen

Het programma in Excel start met de beginsituatie en voert alle verplaatsingen die uit de test komen uit. Wanneer alle verplaatsingen uitgevoerd zijn moet het overzicht in Excel gelijk zijn aan de eindsituatie uit de test. Als dit zo is dan kan gezegd worden dat de test uitvoerbaar is. Voor alle gevonden oplossingen werd de eindsituatie bereikt door de verplaatsingen op de beginsituatie uit te voeren. Dit betekent dat alle oplossingen uitvoerbaar zijn.
Voor de tests die geen oplossing gegenereerd hadden is onderzocht waar het algoritme op vast liep. De drie testen die geen oplossing gegenereerd hebben zijn:

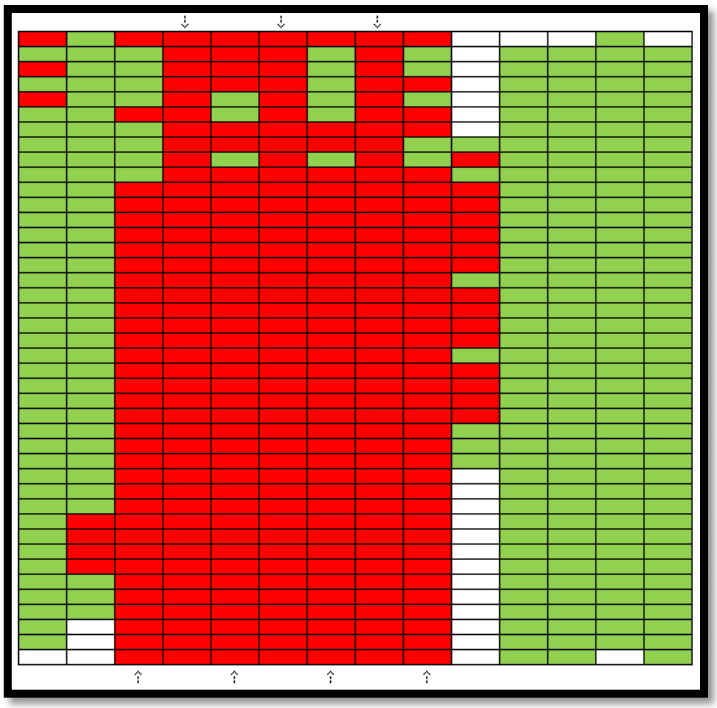
- Test 1: 294 rode rolcontainers (situatie 3)
- Test 2: 211 rode rolcontainers (situatie 2)
- Test 7: 130 rode rolcontainers (situatie 2)

Iedere test zonder oplossing is er één teveel. Daarom moet er naar de oorzaken van het vastlopen van het algoritme gezocht worden. Omdat het algoritme alle mogelijkheden nagaat genereert het algoritme altijd een oplossing. Dat het bij drie tests niet gelukt is komt doordat het algoritme in deze gevallen meer tijd nodig heeft om tot een oplossing te komen. De drie minuten is tekort. Er zijn dan twee opties mogelijk waardoor er alsnog een oplossing gevonden wordt:

- De test langer laten duren
- Het algoritme aanpassen waardoor het sneller werkt

Omdat Anthura Arndt niet lang wil wachten op een oplossing is het langer laten duren van een test een optie die liever niet genomen wordt. Daarom is de tweede optie verder uitgezocht. Het bepalen van verbeteringen die tot een versnelling leiden is gedaan aan de hand van de hierboven genoemde tests. De situatie waar het algoritme op vastliep is geanalyseerd en met de uitkomsten van de analyse is bepaald wat voor aanpassingen er aan het algoritme gedaan moeten worden waardoor het sneller tot een oplossing komt. De analyse met uitkomsten zijn in deze paragraaf uitgeschreven.

Test 1: 294 rode rolcontainers (situatie 3)
Na drie minuten heeft het algoritme geen oplossing kunnen genereren voor deze test. Dit komt omdat na het verplaatsen van 257 rode rolcontainers naar de voorsorteerbanen het algoritme geen verbetering meer vond en vastliep. In figuur 6.3.2.1 is de situatie vanaf waar het algoritme geen verbetering meer vond weergegeven.



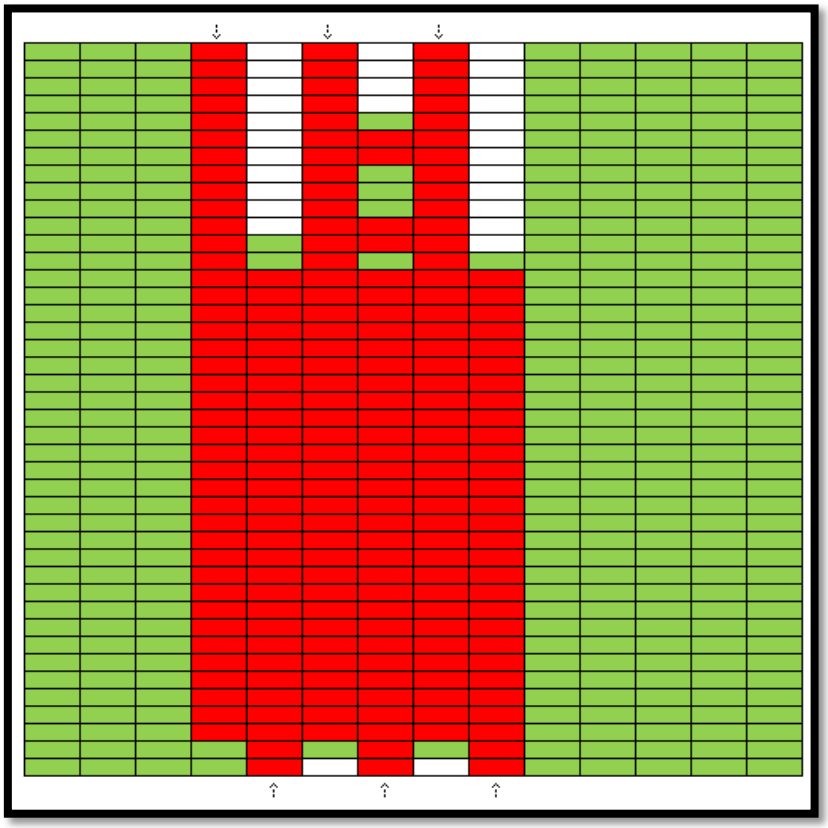
Figuur 6.3.2.1: Situatie bij test 1 waar het programma geen verbetering meer vond.

Vanuit deze situatie wordt nooit meer een oplossing gegenereerd. De reden hiervoor is dat aan alle uiteinden van de voorsorteerbanen rode rolcontainers staan. Omdat een rode rolcontainer alleen een voorsorteerbaan ingeduwd kan worden ontstaat hier een probleem: **De voorsorteerbanen zijn vol.** Er kan dus geen rode rolcontainer de voorsorteerbaan in. Doordat er geen verplaatsingen in of uit de voorsorteerbanen meer mogelijk zijn loopt het programma hierop vast.

De situatie zoals in figuur 6.3.2.1 wordt bereikt wanneer het verplaatsen van een rode rolcontainer vanuit een niet-voorsorteerbaan meer punten oplevert dan een verplaatsing van een rode rolcontainer uit een voorsorteerbaan. Daarnaast is het niet toegestaan een rolcontainer uit een voorsorteerbaan de verplaatsen wanneer deze een lege plek bevat. Hierdoor wordt de mogelijkheid tot verplaatsing te laat gegeven en ontstaat de situatie zoals in figuur 6.3.2.1.

Een verbetering die ervoor kan zorgen dat de situatie als in 6.3.2.1 niet meer voorkomt is het geven van minpunten als een voorsorteerbaan vol is en aan beide kanten een rode rolcontainer staat. Door hier minpunten voor te geven krijgt zo'n situatie het een lagere prioriteit en wordt een verplaatsing die leidt tot zo'n situatie niet gekozen. Wordt er alsnog voor deze situatie gekozen dan wordt dit bij de volgende verplaatsing weer rechtgezet door een van de rode rolcontainers te verplaatsen. De minpunten vervallen dan en daarom is deze verplaatsing aantrekkelijk. De verbetering zorgt er dus voor dat er nooit meer dan één voorsorteerbaan is die vol is met aan beide kanten een rode rolcontainer.

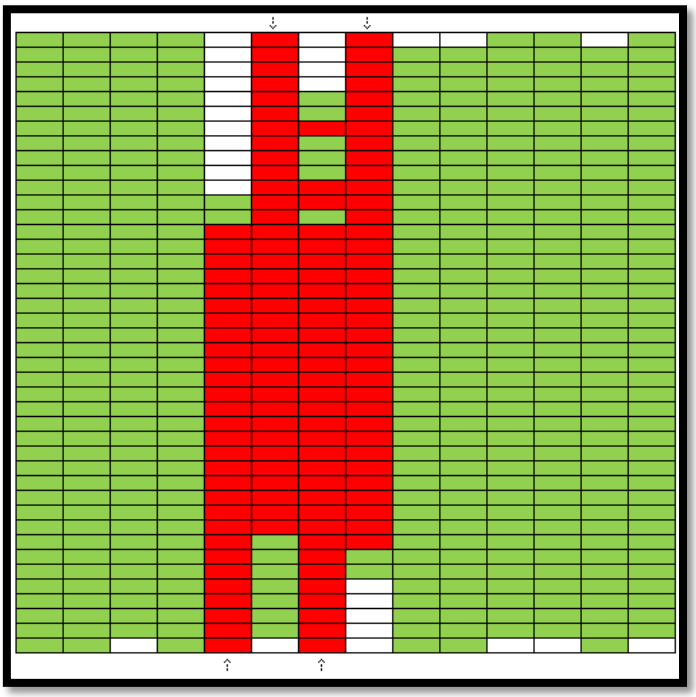
Test 2: 211 rode rolcontainers (situatie 2)
De tweede situatie vond na het verplaatsen van 207 rode rolcontainers geen verbetering meer. In figuur 6.3.2.2 is situatie te zien waar het vastliep.



Figuur 6.3.2.2: Situatie bij test 2 waar het programma op vastloopt.

Het programma loopt in deze situatie vast omdat de niet-voorsorteerbanen allemaal vol zijn. Hierdoor is het niet meer mogelijk groene rolcontainers uit de voorsorteerbanen te halen. In paragraaf 6.1.2 is in fase 3 uitgelegd dat er een groene rolcontainer aan de onderkant van de voorsorteerbaan geplaatst mag worden. Het programma doet dit echter alleen in een voorsorteerbaan met een pijl naar beneden. Te zien is dat er nog twee groene rode rolcontainer in de voorsorteerbanen met een pijl naar beneden geplaatst kunnen worden. Er zijn echter acht groene rolcontainers die dit nog moeten. Dit past niet en daarom komt het programma vanuit deze situatie niet meer verder. Door ook een voorsorteerbaan met een pijl naar boven te nemen om groene rolcontainers in op te slaan wordt het probleem in figuur 6.3.2.2 voorkomen.

Test 7: 130 rode rolcontainers (situatie 2)
De laatste situatie waarbij het programma niet tot een oplossing komt is de tweede situatie uit de zevende test. In figuur 6.3.2.3 is de situatie te zien vanaf het moment dat het programma vastloopt.



Figuur 6.3.2.3: Situatie bij test 7 waar het programma op vastloopt.

In deze situatie krijgen de verplaatsingen die mogelijk zijn een mindere prioriteit dan eerder gedane verplaatsingen. Vanuit deze situatie moet er een groene rolcontainer uit de voorsorteerbaan naar een niet-voorsorteerbaan verplaatst worden. Dit gebeurt niet door de volgende regel:

- Een niet-voorsorteerbaan met minimaal één lege plek levert +40 punten op.

Door het verplaatsen van een groene rolcontainer naar een niet-sorteerbaan krijgt de situatie deze punten niet. Wel worden er meer punten gegeven voor de positie van een rode rolcontainer. Doordat deze extra punten minder zijn dan de veertig uit de regel hierboven krijgt deze situatie een slechtere prioriteit mee en wordt deze verplaatsing niet gekozen.

Een maatregel die dit probleem kan verhelpen is het in fase 2 niet toekennen van punten aan een lege plaats in een niet-voorsorteerbaan. Na deze maatregel krijgt de verplaatsing van een groene rolcontainer wel de voorkeur.

Door deze maatregel komt er een ander probleem naar voren. Een situatie die door een bepaalde verplaatsing van fase veranderd kreeg in fase 1 misschien wel drie honderd punten voor lege plekken. Deze vervallen in fase 2 waardoor de situatie ineens een lager cijfer krijgt. Dit terwijl de situatie in fase 2 dichter bij de oplossing ligt dan de situatie in fase 1.

Door een situatie in fase 2 standaard 14*100 punten te geven wordt dit gat opgevangen en is een situatie in fase 2 bijna altijd beter dan een situatie uit fase 1. Het extra toegekende cijfer wordt het opvulcijfer genoemd.

Tijdsanalyse:

In de tijdsanalyse is gekeken hoeveel procent van de tests een oplossing genereerde binnen drie minuten. In de inleiding van dit hoofdstuk is al beschreven dat er door de lange duur van het testen gekozen is voor het uitvoeren van een tests gedurende drie minuten. Hierna is de test stopgezet. Dit betekent dat alle tests met een oplossing dit binnen drie minuten gegenereerd hebben. Omdat negen van de 171 tests geen oplossing hadden is het percentage dat geen oplossing heeft gegenereerd binnen drie minuten gelijk aan $\frac{9}{171} = 1.713\%$.

Uit dit percentage blijkt dat er bijna aan het tijdsstreven voldaan is. Alleen de negen tests zonder oplossing voldoen niet. Wat niet te zien is aan het percentage is de verdeling van de tijdsduren. Om hier achter te komen is in figuur 6.3.2.4 een frequentietabel te zien. Hierin is weergegeven hoeveel tests er binnen een bepaald aantal seconden een eerste oplossing gegenereerd heeft. De gemiddelde tijdsduur tot de eerste oplossing was 73,86 seconden.

Aantal seconde tot eerste oplossing	Aantal tests
[0 seconde, 30 seconde)	0
[30 seconde, 60 seconde)	51
[60 seconde, 90 seconde)	71
[90 seconde, 120 seconde)	34
[120 seconde, 150 seconde)	6
Meer dan 150 seconde	0

Figuur 6.3.2.4: Frequentietabel van de tijden tot eerste oplossing

Te zien is dat het overgrote deel van de tests tussen de dertig seconden en twee minuten nodig heeft om een oplossing te genereren. Dit is ruim onder de streeftijd. De tests met oplossing hebben daarom voldaan aan de streeftijd.

Efficiëntie-analyse:

Eerder is er een streefafwijking opgesteld. Vastgesteld was dat bij negentig procent van de test de efficiëntieafwijking maximaal twintig procent mocht zijn. Het optimum is echter niet bekend. Door uit te rekenen wat het minimum aantal verplaatsingen is om van beginsituatie tot eindsituatie te komen wordt de ondergrens bepaald. Hierbij wordt uitgegaan van een ideale situatie waarbij er geen rolcontainer zijn die meerdere keren verplaatst moet worden.

Deze ondergrens is voor iedere test berekent. Hierna is per test het verschil berekent tussen de ondergrens en de beste gegenereerde oplossing. Dit verschil wordt de afwijking genoemd. Als laatste wordt dit verschil als een percentage van de ondergrens weergegeven.

De gemiddelde afwijking is 13,55%. In figuur 6.3.2.5 is een frequentietabel te zien met links een interval van percentages en rechts het aantal tests dat zich binnen dit interval bevindt.

Efficiëntieafwijking	Best verkregen oplossing
[0%, 5%)	3
[5%, 10%)	13
[10%, 15%)	21
[15%, 20%)	15
[20%, 25%)	1
[25%, -->)	1

Figuur 6.3.2.5: Frequentietabel

In de tabel is af te lezen dat er 52 tests zijn die een efficiëntieafwijking hebben van minder dan twintig procent. Dit is 96,29% van alle tests. Omdat dit meer dan negentig procent is kan geconcludeerd worden dat het beginalgoritme efficiënt genoeg is.

6.3.3 De conclusie

In deze paragraaf is de conclusie van de hierboven beschreven analyses weergegeven. Per analyse worden de uitkomsten weergegeven.

Uitvoerbaarheidsanalyse:

Na de uitvoerbaarheidsanalyse bleek dat drie van de 57 tests geen oplossing hadden binnen drie minuten. De volgende toevoegingen moeten ervoor zorgen dat deze drie situaties ook een uitkomst krijgen:

- Minpunten geven voor een voorsorteerbaan die vol is en aan beide uiteinden een rode rolcontainer heeft staan
- Een lege plaats op een voorsorteerbaan krijgt in fase twee geen punten meer
- Van te voren krijgt de situatie 14*100 punten. Hierdoor is deze altijd beter dan een situatie uit een eerdere fase
- Er kunnen ook groene rolcontainers aan voorsorteerbanen die naar boven doordraait toegevoegd worden.

Tijdsanalyse:

Alle tests met een uitkomst voordoen aan de streeftijd van drie minuten. Hierdoor heeft het beginalgoritme bijna voldaan aan de streeftijd.

Efficiëntie-analyse:

Uit de efficiëntie-analyse is opgemaakt dat 96,29% van de 54 tests een oplossing had die efficiënt genoeg was. Dit betekent dat de streefafwijking gehaald is. Het algoritme voldoet aan de streefafwijking.

6.4 Aanpassen beginalgoritme

Voordat het beginalgoritme uitgebreid kan worden is het belangrijk dat het algoritme eerst voor alle tests een oplossing genereert. In deze paragraaf wordt beschreven wat er gedaan is zodat alle tests een oplossing genereren.

Allereerst zijn de verbeteringen uit 6.3.2 in het algoritme verwerkt. De volgende verbeteringen worden nog even extra besproken:

- Minpunten voor een voorsorteerbaan die vol is en aan beide uiteinden een rode rolcontainer heeft staan
- Van te voren krijgt de situatie 14*100 punten. Hierdoor is deze altijd beter dan een situatie uit een eerdere fase

Naast het invoeren van de maatregelen uit paragraaf 6.3.3 zijn er een aantal extra onderdelen ingevoerd die het programma gestructureerd maken. Allereerst wordt de volgende factor bij het bepalen van de prioriteit aangepast in fase 2 en verwijdert uit fase 3:

- Positie van een rode rolcontainer aan de voorkant in een voorsorteerbaan

Het aanpassen wordt gedaan omdat er te weinig punten gegeven worden voor de positie van de rode rolcontainer. Hierdoor worden er door het algoritme verkeerde keuzes gemaakt. Door veel testen met verschillende waardes is bepaald dat de vijftig punten in fase 2 verhoogt wordt naar driehonderd punten.

Het verwijderen van de factor uit fase 3 wordt gedaan omdat deze factor ervoor zorgt dat de voorste rode rolcontainers vooraan komen te staan. Omdat het doel van fase 3 het verplaatsen van de groene rolcontainers is, dekt deze factor niet volledig de lading. Daarom wordt het verwijderen van deze factor opgevangen door het invoeren van een nieuwe factor. Bij deze factor worden er minpunten toegekend aan iedere groene rolcontainer die nog in de weg staat. Deze regel wordt ook beschreven.

Als laatste wordt er nog een verbetering toegepast in fase 2 en 3. Er wordt een zogenaamd opvulcijfer ingevoerd. Dit cijfer vervangt het cijfer voor het hebben van een lege plaats op een pijpenbaan.

6.4.1 Minpunten voor een volle voorsorteerbaan:

Bij deze verbetering worden er minpunten gegeven wanneer de voorsorteerbaan vol is met aan beide uiteinden een rode rolcontainer. Per voorsorteerbaan waarbij dit het geval is worden tien punten afgetrokken van het prioriteitencijfer.

6.4.2 Opvulcijfer

Doordat er in iedere fase andere factoren aanwezig zijn komt het voor dat bij de ene fase de factor wel meegenomen wordt terwijl deze in een later fase niet meegenomen wordt. Hierdoor kan het zijn dat een situatie in een latere fase een lager cijfer krijgt omdat er een factor weggevallen is worden en dus minder punten toegekend krijgt. Door aan een latere fase extra punten toe te kennen wordt ervoor gezorgd dat een situatie in een latere fase bijna altijd een hoger cijfer heeft. Hierdoor krijgen latere fases vaker de voorkeur boven eerdere fases. Er zijn meerdere opvulcijfers:

- Opvulcijfer voor lege plaatsen
- Opvulcijfer voor positie van een rode rolcontainer in de voorsorteerbaan

Het opvulcijfer voor de lege plaatsen wordt ingevoerd wanneer er in een fase geen punten meer gegeven worden voor lege plaatsen op pijpenbanen. Eigenlijk is dit een van de maatregelen uit paragraaf 6.3.2 Deze luidde:

- Van te voren krijgt de situatie 14*100 punten. Hierdoor is deze altijd beter dan een situatie uit een eerdere fase.

Het opvulcijfer wordt in fase 2 en 3 ingevoerd. De formule voor het opvulcijfer is als volgt:

$$Opvulcijfer = Totaal\ aantal\ pijpenbanen * +100$$

Bij het opvulcijfer voor de positie van een rode rolcontainer in de voorsorteerbaan wordt er gekeken naar het maximaal aantal punten dat gegeven wordt wanneer deze factor in de fase voorkomt. Het maximaal aantal punten is:

$$Maximaal = Aantal\ voorsorteerbanen * +300$$

Het opvulcijfer voor positie van een rode rolcontainer in de voorsorteerbaan wordt in fase 3 ingevoerd en is gelijk aan het maximum aantal punten:

$$Opvulcijfer = Aantal\ voorsorteerbanen * +300$$

6.4.3 Minpunten voor groene rolcontainers

In de derde fase moeten er alleen nog groene rolcontainers verplaatst worden. Hier komt er een nieuwe factor in het spel. Door minpunten te geven voor iedere groene rolcontainer die nog verplaatst moet worden krijgt zo'n verplaatsing de voorkeur boven een andere verplaatsing.

Het geven van minpunten zorgt ervoor dat de situatie een lager cijfer kan hebben dan een situatie in fase 2. Dit is niet gewenst. Daarom wordt er eerst een soort opvulcijfer aan toegevoegd. Per groene rolcontainer die in de wegstaat krijgt de situatie dertig minpunten. Omdat er maximaal 41 groene rolcontainers voor een rode rolcontainer kunnen staan is het opvulcijfer gelijk aan +1230. De formule voor deze factor is:

$$Punten = 1230 - aantal\ groene\ rolcontainer * 30$$

6.5 Testen van het verbeterde algoritme

Na het invoeren van de extra factoren is het algoritme opnieuw getest. Dit is gedaan met dezelfde tests als in paragraaf 6.3.1. Door dezelfde tests uit te voeren kan het verbeterde algoritme vergeleken worden met het beginalgoritme. De testresultaten bevinden zich in bijlage II. Op de testresultaten zijn ook bij het verbeterde algoritme drie analyses gedaan. Dit zijn de:

- Uitvoerbaarheidsanalyse
- Tijdsanalyse
- Efficiëntie-analyse

6.5.1 Uitvoerbaarheidsanalyse

Uit de uitvoerbaarheidsanalyse is gebleken dat alle tests een oplossing hadden. Daarnaast is leidt de verkregen oplossing na het uitvoeren tot juiste eindsituatie. Het verbeterde algoritme voldoet dus aan de eis die Logiqs B.V. gesteld heeft. Deze luidde:

- Het algoritme moet een oplossing genereren

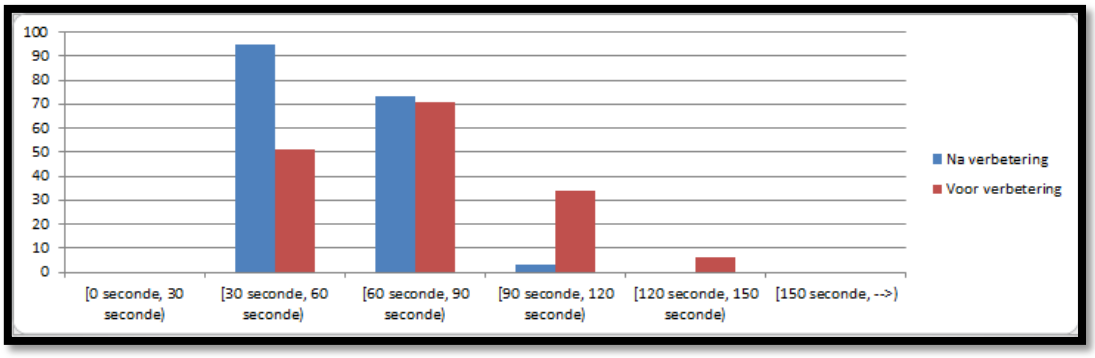
6.5.2 Tijdsanalyse

Over de tijd was geen eis gesteld. Wel was er een streeftijd bepaald waarin gewenst werd een eerste oplossing te genereren. Het beginalgoritme voor de verbetering voldeed al aan deze streeftijd. Hier wordt getest of het verbeterde algoritme ook voldoet aan de streeftijd. In figuur 6.5.1 zijn de resultaten van de tijdsanalyse weergegeven.

Aantal seconde tot eerste oplossing	Aantal tests
[0 seconde, 30 seconde)	0
[30 seconde, 60 seconde)	95
[60 seconde, 90 seconde)	73
[90 seconde, 120 seconde)	3
[120 seconde, 150 seconde)	0
Meer dan 150 seconde	0

Figuur 6.5.1: Resultaten tijdsanalyse

Uit de tabel is af te lezen dat het algoritme ná verbetering voldoet aan de streeftijd. Alle tests hebben binnen drie minuten een oplossing. Vergeleken met het beginalgoritme is ook de snelheid duidelijk verbetert. De gemiddelde tijdsduur na verbetering is 59,52 seconden terwijl dit voor verbetering 73,86 seconden was. Dit is beduidend minder. Om een nog beter vergelijking te maken tussen beide zijn de figuren 6.3.2.4 en 6.5.1 samengevoegd in één grafiek. Deze is hieronder te zien.



Figuur 6.5.2: Grafiek met tijdsduren van voor de verbetering en na de verbetering

In de grafiek is een duidelijke verschuiving naar links te zien. Bij het beginalgoritme waren er nog tijdsduren tot de eerste oplossing die tussen de 120 en 150 seconden lagen. In het verbeterde algoritme is dit niet het geval en bevinden de langste tijdsduren zich tussen de 90 en de 120 seconden. Wat wel overeenkomt is dat er geen tests waren die in minder dan 30 seconden een oplossing gegenereerd hadden. Het algoritme is snel maar niet extreem snel.

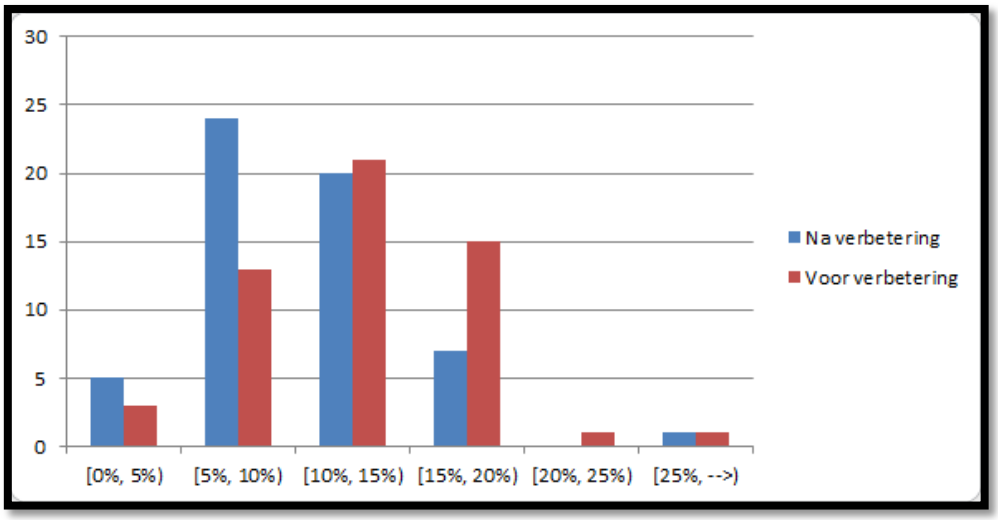
6.5.3 Efficiëntie-analyse:

Ook is er een efficiëntie-analyse gedaan. De resultaten van de analyse staan in figuur 6.5.3.

Efficiëntieafwijking	Best verkregen oplossing
[0%, 5%)	5
[5%, 10%)	24
[10%, 15%)	20
[15%, 20%)	7
[20%, 25%)	0
[25%, -->)	1

Figuur 6.5.3: Resultaten efficiënte-analyse

De gemiddelde efficiëntieafwijking is in dit geval 10,59%. Bij het algoritme vóór de verbetering was dit nog 13,55%. De efficiëntiewijking is flink gedaald. Dit betekent dat het algoritme ná de verbetering efficiënter is. Door ook hier de twee tabellen samen te voegen in een grafiek is nog een keer extra gekeken wat het effect van de verbetering geweest is. De grafiek is te zien in figuur 6.5.4.



Figuur 6.5.4: Grafiek met de efficiëntieafwijking van voor en na de verbetering

In de grafiek is te zien dat er ook hier een verplaatsing naar links is. Het aantal afwijkingen in de linker twee intervallen is gestegen terwijl het aantal afwijkingen in de overige intervallen is gedaald. Er is één uitzondering en dat is de test met de afwijking die groter is dan 25%. De afwijking was voor de verbetering 54%. Na de verbetering is deze nog 34,17%. Deze grote afwijking komt omdat het een speciale test is waarbij het erg moeilijk is de juiste ondergrens te bepalen. Waarschijnlijk is de ondergrens te laag. Een te lage ondergrens resulteert in een te hoge afwijking.

6.5.4 Conclusie:

Er kunnen uit het testrapport twee conclusies getrokken worden. Allereerst kan geconcludeerd worden dat de aanpassing op alle fronten een verbetering geweest is. Zowel de uitvoerbaarheid, de snelheid als de efficiëntie is gestegen. Daarnaast voldoet het verbeterde algoritme aan de gestelde eis, de streeftijd en de streefafwijking. Dit betekent dat het doel van de afstudeeropdracht gehaald is.

7. Het eindalgoritme

De afstudeeropdracht is voltooid. Er is echter nog ruime tijd over om tot een nog beter algoritme te komen. In de resterende tijd wordt er gezocht naar verbeteringen die het algoritme versnellen en efficiënter maken. Daarnaast is het nu mogelijk om extra wensen van Logiqs B.V. in te brengen.

7.1 Verbeteringen algoritme

In een gesprek met de directeur van Logiqs B.V., Gert-Jan van Staalduinen, en mijn begeleider, Koen van Leeuwen, zijn de vervolgstappen bepaalt. Hierbij is vooral gedacht aan verbeteringen voor het algoritme. In deze paragraaf zijn alle verbeteringen die uit dit gesprek voort kwamen beschreven. Deze zijn gerangschikt op prioriteit. Met vooraan de verbetering waar het eerst aan gewerkt wordt. De verbeteringen zijn:

- Het uitvoeren van meerdere strategieën op één startsituatie
- Het invoeren van de Branch and Bound strategie
- Het implementeren van het algoritme in Dat-A-Control
- De graadmeter voor efficiëntieafwijking veranderen
- Het voorsorteren voor meerdere dagen
- Het voorsorteren voor meerdere dagdelen

7.1.1 Het uitvoeren van meerdere strategieën

Het eerste wat bekeken wordt is de mogelijkheid op het verbeteren van de efficiëntieafwijking en duur van het beginalgoritme. Gezocht wordt naar onderdelen in het programma die veranderd kunnen worden waardoor de efficiëntieafwijking en de tijdsduur dalen. Hierbij wordt gedacht aan het uitvoeren van meerdere strategieën achter elkaar.

Op dit moment is er één strategie die uitgevoerd wordt op de startsituatie. Dit is de strategie van het beginalgoritme. Deze strategie is bepaald door het definiëren van de opties en het prioriteitencijfer. Door een tweede strategie te schrijven die op dezelfde startsituatie uitgevoerd wordt ontstaat er een nieuwe route door de zoekboom en dit kan leiden tot een andere oplossing. Deze oplossing is mogelijk efficiënter dan de oplossing uit de eerste strategie.

Bij het ontwerpen van deze verbetering wordt gezocht naar meerdere strategieën die op een andere manier door de zoekboom heen gaan. Deze strategieën worden in het programma geïmplementeerd en getest.

7.1.2 Het invoeren van Branch and Bound

Ook de Branch and Bound is een strategie voor het zoeken naar een betere oplossing dan de oplossing die al gevonden was. Bij de Branch and Bound wordt naar de toekomst gekeken. Voor iedere situatie wordt bepaald wat het minimum aantal verplaatsingen is dat nog nodig is om vanuit de betreffende situatie een oplossing te genereren. Wanneer dit aantal verplaatsingen hoger is dan een eerder gevonden oplossing wordt vanuit deze situatie alleen nog een oplossing gegenereerd die een hogere efficiëntieafwijking heeft dan de oplossing die al gevonden is. Deze situatie wordt dan niet verder uitgewerkt en verwijdert. Door het verwijderen van deze situaties wordt de boom flink verkleind.

7.1.3 Het implementeren van het algoritme

Bij het implementeren van het algoritme in Dat-A-Control wordt het algoritme toegevoegd aan de software van het bedrijf. Na het toevoegen kan via de computer aangegeven worden welke rolcontainers er klaar moeten staan. Hierna wordt er op een knop gedrukt en start het algoritme. De oplossing wordt gelijk ingeladen in het Dat-A-Logiqs en kan hierna uitgevoerd worden.

7.1.4 De graadmeter voor efficiëntie veranderen

Logiqs B.V. wil graag de graadmeter voor de efficiëntie verbeteren. Nu wordt deze bepaald door naar het totaal aantal verplaatsingen in de oplossing te kijken. Logiqs B.V. denkt echter dat het aantal shuttleverplaatsingen een betere graadmeter voor de efficiëntie is dan het aantal rolcontainerverplaatsingen. Zo'n shuttleverplaatsing kan afhankelijk van de afstand die deze moet afleggen zomaar vijf minuten duren. Dit is langer dan de 66 seconden dat een verplaatsing van een rolcontainer duurt. Om deze reden wil Logiqs B.V. de graadmeter voor de efficiëntie veranderen naar het aantal verplaatsingen van de shuttle. Een efficiënte oplossing is nu een oplossing waarbij de shuttle zo weinig mogelijk verplaatst wordt.

7.1.5 Het voorsorteren voor meerdere dagen:

Het doel van de stage was het voorsorteren van rolcontainers voor één dag. Hierdoor zijn er twee verschillende rolcontainers in de kas aanwezig. Dit zijn:

- Rolcontainers die voorgesorteerd moeten worden
- Rolcontainers die niet voorgesorteerd moeten worden

Anthura Arndt heeft nu bedacht dat het misschien wel efficiënter is om in het weekend gelijk voor de hele week voor te sorteren. Dit betekent dat er niet twee maar zes verschillende rolcontainers in de kas aanwezig zijn. Dit zijn:

- Rolcontainers die voorgesorteerd moeten worden voor maandag
- Rolcontainers die voorgesorteerd moeten worden voor dinsdag
- Rolcontainers die voorgesorteerd moeten worden voor woensdag
- Rolcontainers die voorgesorteerd moeten worden voor donderdag
- Rolcontainers die voorgesorteerd moeten worden voor vrijdag
- Rolcontainers die niet voorgesorteerd moeten worden

Het probleem wordt hierdoor wat gecompliceerder. De opties die er zijn veranderen en daarbij moeten er ook punten gegeven worden voor het verplaatsen van rolcontainers die bestemd zijn voor andere dagen.

7.1.6 Het voorsorteren voor meerdere dagdelen

Als laatste ziet Logiqs B.V. er een verbetering in om niet alleen op dag maar ook op het dagdeel te sorteren. Het idee is dat de rolcontainers met orders voor s' ochtends vooraan staan en dat de rolcontainers met orders voor de middag hierachter staan. Ook door deze verbetering wordt het probleem gecompliceerder. Er is namelijk een zekere volgorde van rolcontainers.

7.2 Eerste verbetering

De eerste verbetering moet ertoe leiden dat er een betere oplossing gevonden wordt. Hierbij is de tijd tot een oplossing niet van belang. Natuurlijk is het de wens om zo snel mogelijk tot een oplossing te komen, maar er is geen eis voor gesteld. Wel is er een eis gesteld aan de efficiëntieafwijking. Eerder was de streefafwijking als volgt opgesteld:

- Het streven is dat bij negentig procent van de oplossingen de streefafwijking hoogstens twintig procent is.

De nieuwe streefafwijking bestaat uit twee onderdelen. Allereerst wordt er gestreefd naar een gemiddelde efficiëntieafwijking van minder dan tien procent. Daarnaast wordt gestreefd naar een efficiëntieafwijking van minder dan vijf procent voor twintig procent van de tests. Dit komt neer op twaalf tests waarbij de efficiëntieafwijking minder dan vijf procent moet zijn.

Het bereiken van deze streefafwijking wordt gedaan door meerdere strategieën toe te passen op de startsituatie. Daarnaast wordt de Branch and Bound strategie ingevoerd. De Branch and Bound strategie zorgt ervoor dat er bij het uitwerken van een situatie gekeken wordt naar de beste oplossing die tot dan toe gevonden is. Heeft de situatie een slechter vooruit dan wordt deze verwijdert. Hierdoor worden er niet onnodig situaties onderzocht.

7.2.1 Gebruiken van meerdere strategieën

Eerder kwam naar voren dat het aanpassen van factoren leidt tot een andere route door de zoekboom. In deze paragraaf is beschreven welke factoren er aangepast kunnen worden zodat de route veranderd.

Eigenlijk betreft het alle factoren die bij het bepalen van de prioriteit meetellen. Iedere factor kan dusdanig veranderd worden dat er een andere route door de zoekboom ontstaat. Helaas betekent dat niet altijd dat het een gunstige andere route is. Het veranderen van een aantal factoren kan ertoe leiden dat het algoritme geen oplossing meer genereerd binnen de drie minuten dat de test duurt.

Allereerst is gekeken of het mogelijk is een factor te verwijderen. Door om de beurt een factor te verwijderen en hierna de tests weer uit te voeren is getest of dit een mogelijk extra strategie is. Hieruit bleek dat alleen de factor waarbij er punten geven werden aan een lege plek op een niet-voorsorteerbaan verwijdert kan worden. Verderop in deze paragraaf zijn de testresultaten weergegeven.

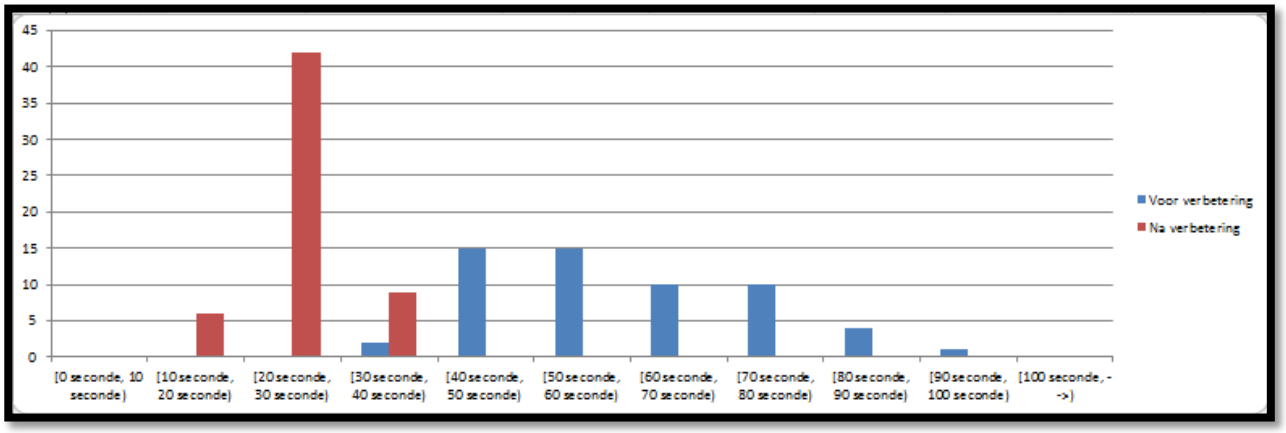
Verder is per factor bekeken of deze aangepast kan worden zodat er een andere route ontstaat. Hierbij is alleen gekeken naar de factoren in de eerste fase. Dit omdat deze het meeste effect hebben op de route door de zoekboom. In fase twee en drie zijn minder verplaatsingen mogelijk. Minder verplaatsingen betekent dat de factoren op maar minder verplaatsingen invloed hebben. De route verandert hierdoor bijna niet. Daarnaast wordt door de huidige factoren de beste route in fase 2 en fase 3 bijna altijd gevolgd. Daarom valt er in deze fases weinig winst te behalen. De factoren uit fase 1 die onderzocht worden zijn:

- Punten voor een lege plaats op een voorsorteerbaan
- Punten voor de positie van een rode rolcontainer op een niet-voorsorteerbaan
- De vermenigvuldigingsfactor voor de positie van een rode rolcontainer op een niet-voorsorteerbaan
- Punten voor een rode rolcontainer op een voorsorteerbaan

Een extra factor die kan leiden tot een betere oplossing is het invoeren van minpunten per uitgevoerde verplaatsing. De factor is in het beginalgoritme niet aanwezig. In deze paragraaf wordt het effect van deze factor beschreven.

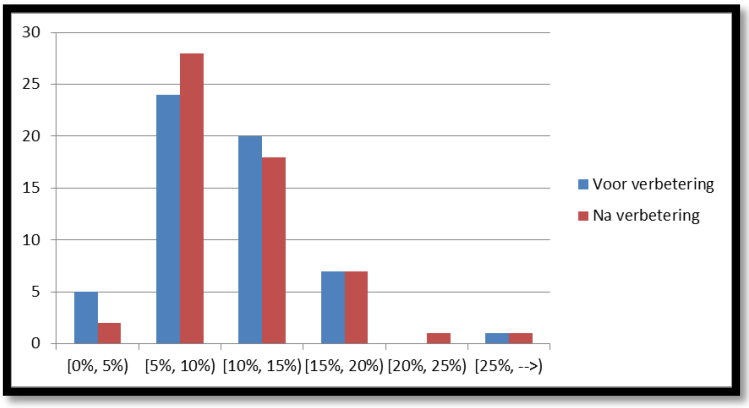
Verwijderen van de punten voor lege plek op niet-voorsorteerbaan:

Eerder is beschreven dat het mogelijk was om de punten voor een lege plek op een niet-voorsorteerbaan te verwijderen. Na het verwijderen van de factor heeft iedere test nog steeds een oplossing. Getest is wat het effect van het verwijderen van deze punten heeft op de efficiëntieafwijking en tijdsduur van het algoritme. De testresultaten over de tijdsduur en de efficiëntieafwijking zijn in de figuren 7.2.1.1 en 7.2.1.2 weergegeven.



Figuur 7.2.1.1: Testresultaten over de tijdsduur

Uit figuur 7.2.1 is op te maken dat het verwijderen van deze factor een grote invloed heeft op de tijd die het kost tot het genereren van de eerste oplossing. De gemiddelde tijdsduur was 59,52 seconden en is na het verwijderen van de factor nog maar 25,70 seconden. De tijdsduur is meer dan gehalveerd.



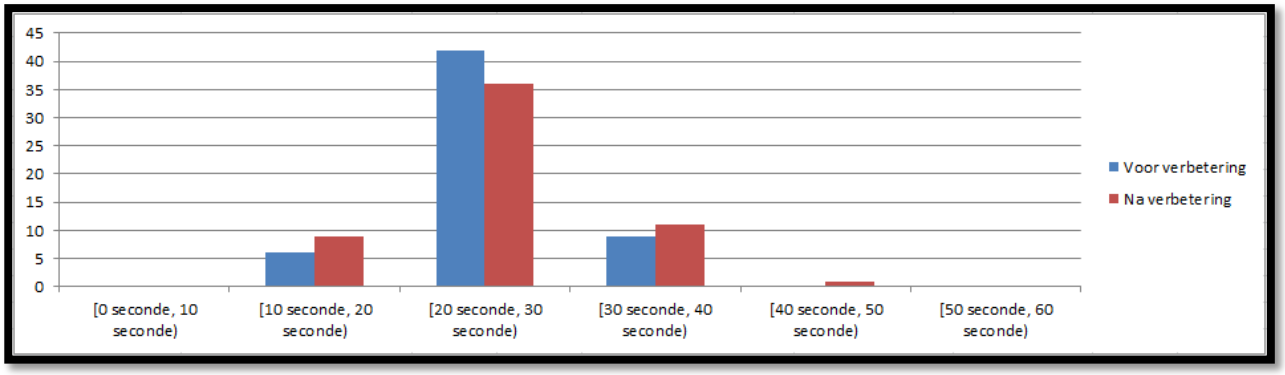
Figuur 7.2.1.2: Testresultaten over de efficiëntieafwijking

In figuur 7.2.1.2 is te zien dat het verwijderen van deze factor ook van lichte invloed is op de efficiëntieafwijking. De gemiddelde efficiëntieafwijking was vóór de verbetering 10,99% en is na de verbetering gedaald naar 10,59%. Er is dus een lichte verbetering in de efficiëntieafwijking.

Omdat het verwijderen van deze factor het programma flink versneld is ervoor gekozen de verbetering in alle strategieën in te voeren en hierna te zoeken naar andere verbeteringen.

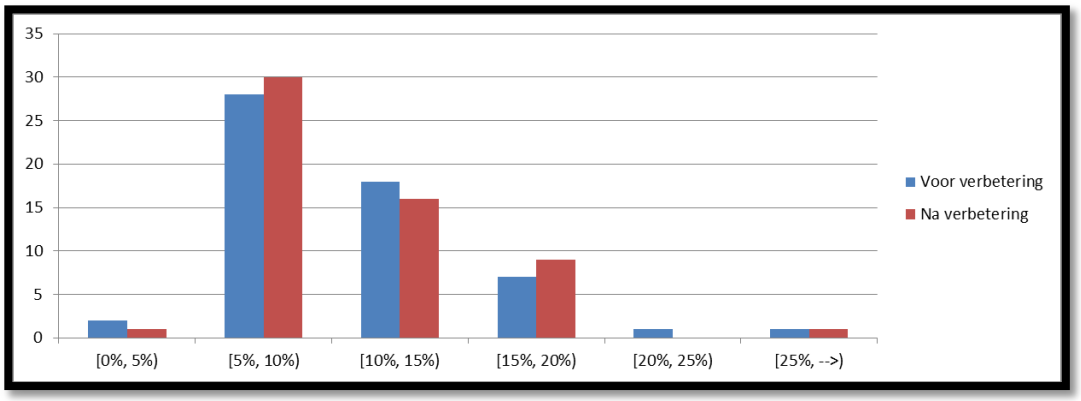
Punten voor de positie van een rode rolcontainer:

In fase 1 worden er punten gegeven voor de positie van een rode rolcontainer. Het maximum aantal punten dat hier gegeven wordt is vijftig. Door deze eerst te verlagen en daarna te verhogen is bekeken wat het effect is op de uitkomst. De resultaten voor het verlagen zijn weergegeven in de figuren 7.2.1.3 en 7.2.1.4.



Figuur 7.2.1.3: Testresultaten over de tijdsduur

De gemiddelde tijdsduur tot de eerste oplossing is door deze verandering bijna niet veranderd. De gemiddelde tijdsduur was 25,70 seconden en is na de verandering 25,45 seconden. Uit het figuur is ook op te maken dat er niet veel veranderd is.



Figuur 7.2.1.4: Testresultaten over de efficiëntieafwijking

Ook de efficiëntieafwijking veranderd niet erg veel. De gemiddelde efficiëntieafwijking was 10,59%. Dit is na de verbetering 10,38%. Uit deze resultaten kan geconcludeerd worden dat het geen grote verbetering is. Maar omdat er wel verschillen zijn in de efficiëntieafwijking wordt door deze verandering een andere route door de boom genomen wordt. Hierdoor kan het wel als één van de strategieën gekozen worden.

Ook het verhogen van de factor is getest. Uit deze tests bleek dat de gemiddelde efficiëntieafwijking door de verhoging gedaald is van 10,59% naar 10,55%. Dit is een minimale verbetering. De gemiddelde tijdsduur tot de eerste oplossing is van 25,70 naar 24,11 seconden gedaald. Met anderhalve seconden verschil is dit wel een grote verbetering. Ook de verhoging van de factor is een mogelijkheid om als strategie te gebruiken.

Punten voor een rode rolcontainer op een voorsorteerbaan:

Ook is het effect bekeken van het aanpassen van de punten voor een rode rolcontainer op een voorsorteerbaan. Hierbij is getest wat er gebeurt als het aantal punten veranderd naar tweehonderd en zeshonderd. Uit de tests blijkt dat deze verandering het algoritme allereerst vertraagd. Daarnaast is na het verhogen de oplossing bij 55 van de 57 test exact hetzelfde.

Het verlagen van de factor zorgt ervoor dat één van de 57 tests niet binnen drie minuten een oplossing genereert. Dit terwijl deze test voor de aanpassing in vijftien seconden een oplossing genereerde. Daarnaast is ook de gemiddelde efficiëntieafwijking hoger dan vóór de verandering.

Geconcludeerd kan worden dat het aanpassen van de punten voor de rode rolcontainers geen goede verbetering is. Daarom wordt zowel het verhogen als het verlagen van de factor niet als strategie gebruikt.

Vermenigvuldigingsfactor voor de positie van een rode rolcontainer:

Door de vermenigvuldigingsfactor te verlagen en daarna te verhogen is getest wat het effect van een verandering in deze factor is. Wanneer blijkt dat er een verschil ontstaat in de route door de zoekboom dan is het veranderen van deze factor een strategie die gebruikt kan worden bij het verbeteren van het algoritme.

Het verlagen van de factor levert 55 andere oplossingen op. Dit betekent dat bij 96,5% van de tests de zoekboom op een andere manier doorlopen wordt. De tijd die het kost om tot deze oplossingen te komen veranderd bijna niet.

Het verhogen van de factor resulteert in vijftig andere oplossingen. Dat betekent dat 87,72% van de tests een andere route door de zoekboom gevolgd heeft. Daarnaast is ook de gemiddelde afwijking met 0,48% gedaald.

Zowel het verhogen als verlagen van deze factor levert een andere route door de zoekboom op. Beide kunnen in het verbeterde algoritme als strategie gebruikt worden.

Punten voor een lege plaats op een voorsorteerbaan:

Het verlagen of verhogen van de factor levert voor maar een paar testen een andere route op. Het overgrote deel van de tests bewandeld dezelfde route en heeft dus ook dezelfde uitkomst. Om deze reden is ervoor gekozen deze factor niet te veranderen.

Minpunten geven voor iedere verplaatsing die uitgevoerd is:

De inspiratie voor deze factor komt uit het filmpje van Leon Bouquiet. Door voor iedere verplaatsing die uitgevoerd is een aantal minpunten te geven wordt er niet te snel de diepte van de zoekboom in gedoken. Hoe dit precies werkt wordt uitgelegd met een voorbeeld.

Stel dat er twee onthouden situaties zijn. Beide situaties hadden dezelfde beginsituatie. De eerste situatie heeft:

- Honderd rode rolcontainers op de voorsorteerbanen
- Een prioriteitencijfer van duizend
- Tweehonderd verplaatsingen uitgevoerd.

De tweede situatie heeft:

- Negentig rode rolcontainers op de voorsorteerbanen
- Een prioriteitencijfer van negenhonderd
- Honderd verplaatsingen uitgevoerd.

De eerste situatie heeft het hoogste cijfer en wordt daarom gekozen om als eerste uit te werken. Eerder is gezegd dat een situatie met een hoog cijfer dichter bij de oplossing is dan een situatie met een laag cijfer. Er was namelijk geëist dat er een oplossing gegenereerd moest worden. De efficiëntie hiervan maakte niet uit. Nu ook de efficiëntie meegenomen wordt klopt de regel eigenlijk niet meer. De eerste situatie heeft gemiddeld twee verplaatsingen nodig om één rode rolcontainer naar een voorsorteerbaan te verplaatsen en de tweede situatie heeft gemiddeld 1,11 verplaatsingen nodig om dit te doen. Hiernaar gekeken is eigenlijk de tweede situatie efficiënter.

Door voor iedere uitgevoerde verplaatsing twee minpunten te geven veranderen de prioriteit cijfers. De eerste situatie krijgt hierna het cijfer zeshonderd. De tweede situatie krijgt het cijfer zevenhonderd. Het cijfer van situatie twee is nu hoger den wordt hierdoor gekozen om verder uit te werken. Gekeken naar de efficiëntie klopt dit beter.

Het invoeren van de minpunten zorgt ervoor dat het programma langer nodig heeft om tot een oplossing te komen. Doordat een efficiëntere oplossing belangrijker is dan de tijdsduur maakt dit niet uit.

Conclusie:

Uit de bovenstaande paragraaf kan geconcludeerd worden dat het veranderen van een aantal factoren leidt tot een andere route door de zoekboom. Bij sommige factoren is dit niet het geval. De factoren waarbij een verandering effect heeft op de route zijn:

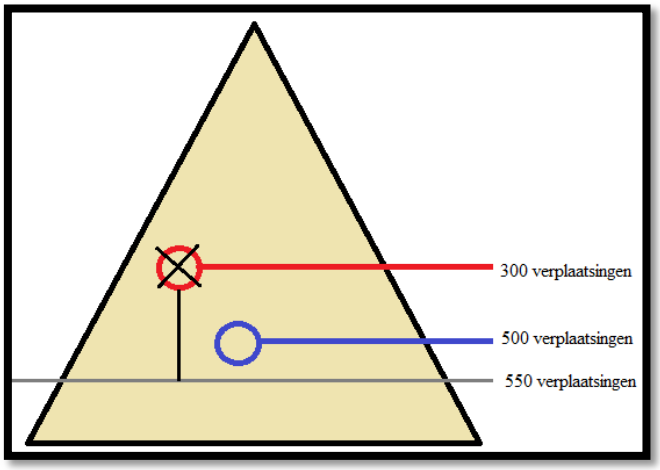
- Minpunten geven voor iedere verplaatsing die uitgevoerd is:
- Vermenigvuldigingsfactor voor de positie van een rode rolcontainer:
- Punten voor de positie van een rode rolcontainer:

7.2.2 Invoeren Branch and Bound

In deze paragraaf is beschreven wat het Branch and Bound principe inhoudt. Het Branch and Bound principe zegt dat het geen zin heeft een tak uit te werken wanneer van te voren al bepaald is dat de efficiëntste oplossing die uit de tak kan voortkomen, een grotere efficiëntieafwijking heeft dan een oplossing die al gevonden is. Met een voorbeeld wordt dit uitgelegd.

Stel er is een algoritme dat een oplossing gegenereerd heeft en er wordt gezocht naar een nieuwe oplossing. De gevonden oplossing werd bereikt door vijfhonderd verplaatsingen uit te voeren. Nu komt het algoritme in een situatie waarin er driehonderd verplaatsingen uitgevoerd zijn. Normaal gesproken wordt deze situatie gewoon doorgerekend. Maar nu is ook uitgerekend dat vanuit deze situatie de kortste weg naar een oplossing gelijk is aan 250 verplaatsingen. Dat betekent dat de beste oplossing in deze tak neerkomt op een oplossing met 550 verplaatsingen. Dit is meer dat de eerder gevonden oplossing. Het uitwerken van deze situatie levert geen betere oplossing op. Daarom is het beter deze situatie niet uit te werken.

In figuur 7.2.2.1 is een driehoek weergegeven. Deze stelt de zoekboom voor. Het blauwe rondje is de oplossing die gevonden is en het rode rondje is de situatie waarin het algoritme zich nu bevind. Vanuit het rode rondje is een lijn naar beneden getekend. Deze komt onder het blauwe rondje uit en levert daarom een oplossing op met een grotere efficiëntieafwijking.



Figuur 7.2.2.1: Driehoek met het concept weergegeven

Voordat deze nieuwe toepassing effect heeft moeten er twee onderdelen aanwezig zijn. Allereerst moet het algoritme al een oplossing gevonden hebben. Wanneer deze er niet is kan deze toepassing niet gebruikt worden. Ten tweede moet er voor iedere situatie uitgerekend worden wat de kortste weg is om vanuit de situatie een oplossing te genereren.

Bij het testen in paragraaf 6.3 is beschreven dat de efficiëntieafwijking van de oplossingen berekend wordt door het verschil te berekenen tussen de gevonden oplossing en de ondergrens. Deze ondergrens wordt ook gebruikt voor het bepalen van het minimaal aantal verplaatsingen. Bij het aanmaken van nieuwe situaties wordt berekent wat de ondergrens van de betreffende situatie is. Deze wordt vergeleken met de eerste oplossing. Is de ondergrens lager dan het aantal verplaatsingen uit de oplossing dan wordt deze oplossing weggegooid.

7.2.3 Testen eerste verbetering

Voor het testen van de verbetering zijn dezelfde tests gebruikt als in paragraaf 6.3. Op deze tests zijn meerdere strategieën uitgevoerd. Deze strategieën zijn bepaald aan de hand van de uitkomsten uit het paragraaf 7.2.1. Iedere test wordt uitgevoerd tot deze een oplossing gegenereerd heeft. Na het genereren van een oplossing wordt de volgende test uitgevoerd. Wanneer een test geen oplossing genereerd wordt deze tien seconden nadat er geen verbetering gevonden wordt afgebroken. Ook nu wordt de volgende test gestart. Er zijn drie factoren die aangepast worden. Hieronder zijn de factoren met de veranderde waardes weergegeven.

- Vermenigvuldigingsfactor = 0,7
- Vermenigvuldigingsfactor = 0,8
- Vermenigvuldigingsfactor = 0,9
- Vermenigvuldigingsfactor = 0,95
- Vermenigvuldigingsfactor = 0,99
- Minpunten per rolcontainerverplaatsing = 1 per verplaatsing
- Minpunten per rolcontainerverplaatsing = 0 per verplaatsing
- Max punten positie rode rolcontainer = 49
- Max punten positie rode rolcontainer = 50
- Max punten positie rode rolcontainer = 55

Met deze factoren zijn alle mogelijke combinaties gemaakt. De combinaties zijn in de komende paragraaf weergegeven als (Minpunten per rolcontainerverplaatsing; Vermenigvuldigingsfactor; Max punten positie rode rolcontainer).

(1; 0,95; 50)	(1; 0,90; 49)	(1; 0,8; 49)
(0; 0,95; 50)	(0; 0,90; 49)	(0; 0,8; 49)
(1; 0,99; 50)	(1; 0,95; 55)	(1; 0,7; 55)
(0; 0,99; 50)	(0; 0,95; 55)	(0; 0,7; 55)
(1; 0,90; 50)	(1; 0,99; 55)	(1; 0,8; 55)
(0; 0,90; 50)	(0; 0,99; 55)	(0; 0,8; 55)
(1; 0,95; 49)	(1; 0,90; 55)	(1; 0,7; 50)
(0; 0,95; 49)	(0; 0,90; 55)	(0; 0,7; 50)
(1; 0,99; 49)	(1; 0,7; 49)	(1; 0,8; 50)
(0; 0,99; 49)	(0; 0,7; 49)	(0; 0,8; 50)

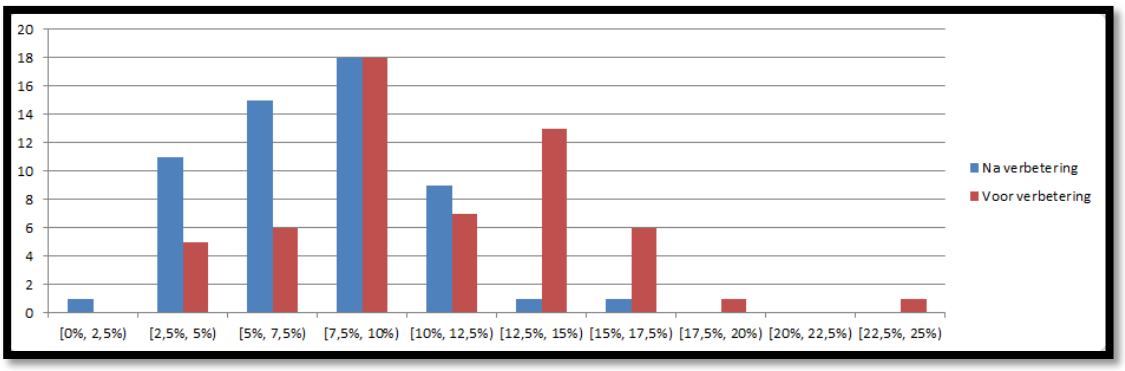
Figuur 7.2.3.1: De strategieën

Na de tests wordt bekeken welke strategieën geen of bijna geen verbeterde oplossingen opleveren. Deze tests worden verwijderd.

Efficiëntie-analyse:

Met de uitkomsten van de tests is er een efficiëntie-analyse gedaan. Hierbij is er per test gekeken naar de beste oplossing uit de dertig strategieën. Voor deze tests is de afwijking in percentages van de ondergrens berekend. De afwijking is in figuur 7.2.3.2 visueel gemaakt. Naast de afwijkingen van ná de verbetering zijn ook de afwijkingen van vóór de verbetering weergegeven. Het algoritme van vóór de verbetering is het beginalgoritme.

Figuur 7.2.3.2: Visueel beeld van het effect van de verbetering.



In de grafiek is een verschuiving naar links te zien. Dit betekent dat extra strategieën de efficiëntieafwijking van de oplossingen verlaagt. Dit is ook te zien in de gemiddelde efficiëntieafwijking. De gemiddelde efficiëntieafwijking was vóór de verbetering 10,59%. Na de verbetering is dit 7,72%. Dit is een verbetering van meer dan drie procent. Het invoeren van meerdere strategieën heeft dus een positief effect.

Analyse over de oplossingen:

Ook is er per strategie bekeken of er een test was waarvoor deze de beste oplossing genereerde gedurende het proces. Hieruit bleek dat er acht strategieën waren die geen enkele keer de beste oplossing gegenereerd hadden. De strategieën zijn:

- (1; 0,99; 50)
- (1; 0,99; 49)
- (1; 0,99; 55)
- (0; 0,8; 55)
- (1; 0,7; 50)
- (0; 0,7; 50)
- (1; 0,8; 50)
- (0; 0,8; 50)

Hieruit blijkt dat de combinatie tussen een factor van 0,99 en één minpunt per verplaatsing geen oplossing oplevert. De strategieën met deze combinatie worden verwijdert. Het verlagen van de vermenigvuldigingsfactor levert bij de meeste strategieën ook geen oplossing op. Ook deze worden verwijderd. Het verhogen van de factor leverde juist wel betere oplossingen op. Daarom is gekozen om de vermenigvuldigingsfactor nogmaals aan te passen en te testen. De factor is aangepast naar 0,975. De strategieën die hieruit volgden zijn:

- (1; 0,975; 50)
- (0; 0,975; 50)
- (1; 0,975; 55)
- (0; 0,975; 55)
- (1; 0,975; 49)
- (0; 0,975; 49)

Ook de nieuwe strategieën zijn getest. De testen leverden voor acht test een betere oplossing op. Door deze verbetering is de gemiddelde afwijking gedaald naar 7,65%. Deze verbeteringen zijn voortgekomen uit drie van de zes strategieën. De andere strategieën hebben geen oplossing gegenereerd. Deze drie strategieën worden verwijdert. De strategieën zijn:

- (1; 0,975; 50)
- (1; 0,975; 55)
- (1; 0,975; 49)

Als laatste is gebleken dat de strategie die als eerste uitgevoerd wordt voor vijf tests geen oplossing genereerde. Gewenst is dat de eerste strategie voor alle tests een oplossing genereert. Daarom is besloten een strategie vooraan te zetten die altijd een oplossing genereert. De strategie is:

- (0; 0,95; 50)

Conclusie:

Uit de efficiëntie-analyse is geconcludeerd dat het gebruik van meerdere strategieën een positief effect van bijna drie procent heeft op de efficiëntieafwijking.

Uit de analyse op de oplossingen kan geconcludeerd worden dat niet alle strategieën meewerken aan dit positieve effect. Sommige strategieën genereren helemaal geen oplossing. Dit geldt voor de strategieën met een vermenigvuldigingsfactor van 0,99 en 0,975 in combinatie met één minpunt per verplaatsing. Daarnaast kan ook geconcludeerd worden dat een strategie met een lage vermenigvuldigingsfactor meestal geen oplossing genereert.

7.2.4 Optimalisatie strategieën

Door de strategieën na elkaar uit te voeren is gekeken welke strategie de meeste oplossingen genereerde. Na het testen is de strategie met de meeste oplossingen vooraan gezet en de strategie met de minste oplossingen achteraan. In de tabel hieronder is de volgorde van het uitvoeren van de strategieën te zien.

1. (0; 0,95; 50)	10. (1; 0,95; 55)	18. (1; 0,8; 49)
2. (1; 0,95; 50)	11. (0; 0,95; 55)	19. (0; 0,8; 49)
3. (0; 0,99; 50)	12. (0; 0,99; 55)	20. (1; 0,7; 55)
4. (1; 0,90; 50)	13. (0; 0,99; 55)	21. (0; 0,7; 55)
5. (0; 0,90; 50)	14. (1; 0,90; 55)	22. (1; 0,8; 55)
6. (1; 0,95; 49)	15. (0; 0,90; 55)	23. (0; 0,975; 50)
7. (0; 0,95; 49)	16. (1; 0,7; 49)	24. (0; 0,975; 55)
8. (0; 0,99; 49)	17. (0; 0,7; 49)	25. (0; 0,975; 49)
9. (1; 0,90; 49)		

Figuur 7.2.4.1: Strategieën die getest zijn

Uit de testresultaten is bepaald wat de nieuwe volgorde van de strategieën is. In de tabel hieronder is de nieuwe volgorde van de strategieën te zien.

1. (0; 0,95; 50)	10. (1; 0,8; 49)	18. (0; 0,95; 49)
2. (1; 0,95; 50)	11. (1; 0,90; 49)	19. (1; 0,90; 55)
3. (0; 0,99; 50)	12. (0; 0,99; 55)	20. (1; 0,7; 49)
4. (0; 0,975; 50)	13. (0; 0,975; 55)	21. (0; 0,7; 49)
5. (1; 0,95; 49)	14. (0; 0,975; 49)	22. (0; 0,95; 55)
6. (1; 0,95; 55)	15. (0; 0,8; 49)	23. (0; 0,90; 55)
7. (0; 0,99; 49)	16. (1; 0,7; 55)	24. (0; 0,7; 55)
8. (1; 0,90; 50)	17. (0; 0,90; 50)	25. (1; 0,8; 55)
9. (0; 0,90; 49)		

Figuur 7.2.4.2: Strategieën op volgorde gezet

7.3 Tweede verbetering

Als tweede verbetering is de implementatie doorgevoerd. Deze implementatie is in een halve dag gerealiseerd en is gedaan met hulp van Koen van Leeuwen en Maarten van der Veen. Zij hebben een Excel bestand gemaakt dat als input voor het algoritme werkt. Het algoritme start na deze invoer van de input. Wanneer het algoritme klaar is geeft deze de output terug. Koen en Maarten hebben het programma aangepast zodat de gekregen output in Dat-A-Logiqs ingeladen wordt. Hierna worden de verplaatsingen één voor één uitgevoerd op het kasoverzicht en is meteen de indeling te zien die ontstaat na het uitvoeren van de verplaatsingen. In deze paragraaf is het volgende beschreven:

- De input voor het algoritme
- De output van het algoritme

7.3.1 De input voor het algoritme

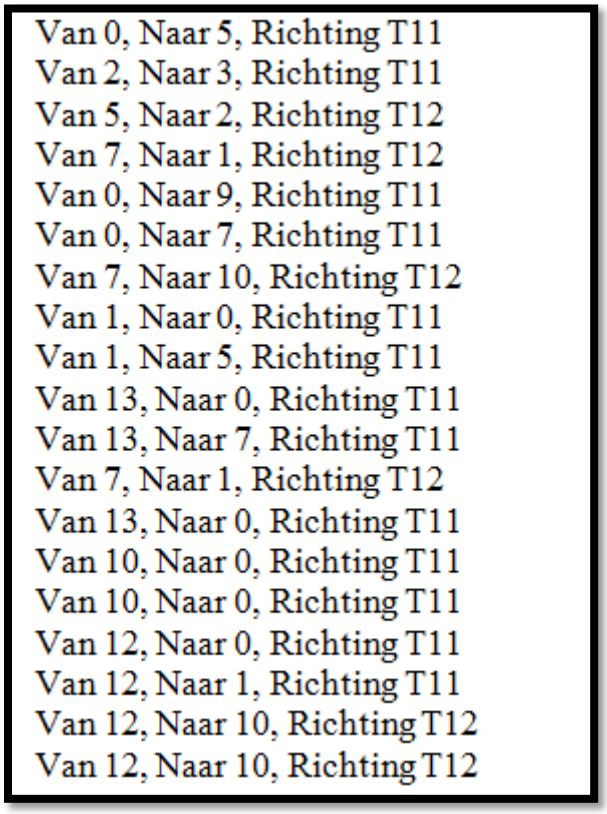
De input voor het algoritme is een Excel bestand dat ingeladen wordt. In dit bestand zijn zeven kolommen belangrijk. In deze zeven kolommen kan per dag van de week ingevoerd worden welke rolcontainers er op die dag voorgesorteerd moeten worden. Wanneer dit ingevuld is en de tuinder het bestand inlaad gaat het programma de rolcontainers die bij maandag staan voorsorteren. In figuur 7.3.1 is een deel van het Excel bestand te zien. In de kolom met ‘Maandag’ staan de nummers van de voor te sorteren rolcontainers.

	A	B	C	D	E	F	G
1	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag	Zondag
2	101						
3	201						
4	501						
5	325						
6	541						
7	154						
8	165						
9	198						
10	265						
11	543						
12	254						
13							
14							

Figuur 7.3.1: Excel bestand dat als input voor het programma geldt

7.3.2 De output van het algoritme

De output van het programma bestaat uit een kladblokbestand met regels. Iedere regel stelt een verplaatsing voor. Hierbij is vermeld vanuit welke pijpenbaan er een rolcontainer verplaatst moet worden en naar welke pijpenbaan deze moet. Ook staat in deze regel of de rolcontainer T11 of T12 verplaatst moet worden. In figuur 7.3.2 is de output van het programma te zien.


Van 0, Naar 5, Richting T11
Van 2, Naar 3, Richting T11
Van 5, Naar 2, Richting T12
Van 7, Naar 1, Richting T12
Van 0, Naar 9, Richting T11
Van 0, Naar 7, Richting T11
Van 7, Naar 10, Richting T12
Van 1, Naar 0, Richting T11
Van 1, Naar 5, Richting T11
Van 13, Naar 0, Richting T11
Van 13, Naar 7, Richting T11
Van 7, Naar 1, Richting T12
Van 13, Naar 0, Richting T11
Van 10, Naar 0, Richting T11
Van 10, Naar 0, Richting T11
Van 12, Naar 0, Richting T11
Van 12, Naar 1, Richting T11
Van 12, Naar 10, Richting T12
Van 12, Naar 10, Richting T12

Figuur 7.3.2: Output van het programma

Koen en Maarten hebben een stukje code geschreven dat deze regels in het transportplan laadt. Na het inladen worden de verplaatsingen uitgevoerd en is te zien wat de kasindeling na het verplaatsen wordt.

7.3.3 Conclusie

Omdat de implementatie van het algoritme binnen een dag gedaan is kan geconcludeerd worden dat de input en de output van het algoritme het gewenste format had. De koppeling tussen de software en Dat-A-Logiqs hebben ze daarom erg snel kunnen maken.

7.4 Derde verbetering

Als derde verbetering wordt de graadmeter voor de efficiëntie vastgesteld op het aantal shuttleverplaatsingen. Een efficiënte oplossing is nu een oplossing met een laag aantal shuttleverplaatsingen. Er zijn een aantal aanpassingen gedaan om dit door te kunnen voeren. Dit zijn:

- Geen ondergrens meer
- Shuttleverplaatsingen als graadmeter
- De minpunten per shuttleverplaatsing

De uitkomst van deze verbetering is een reeks strategieën waarbij het aantal shuttleverplaatsingen als graadmeter genomen wordt. De uitkomst wordt hierna op dezelfde manier geoptimaliseerd als in paragraaf 7.2.4.

7.4.1 Geen ondergrens meer

Bij het berekenen van een ondergrens wordt eigenlijk een toekomstvoorspelling gedaan. Voor het berekenen van het aantal shuttleverplaatsingen is het nodig te weten welke verplaatsingen er uitgevoerd moeten worden. Omdat deze voor de toekomst niet bepaald zijn is het niet mogelijk een toekomstvoorspelling te maken voor het aantal shuttleverplaatsingen. Hierdoor kan er geen ondergrens bepaald worden en is het niet mogelijk om situaties te verwijderen omdat verwacht wordt dat deze een oplossing met een grotere efficiëntieafwijking gaan genereren. Wel kan er berekent worden hoeveel shuttleverplaatsingen er nodig zijn voor het uitvoeren van de verplaatsingen die al bekend zijn. Wanneer dat aantal hoger ligt dan de tot dan toe beste oplossing wordt de situatie wel verwijdert.

7.4.2 Shuttleverplaatsingen als graadmeter

Eerder werd een oplossing efficiënt genoemd als deze een laag aantal verplaatsingen van rolcontainers nodig had om tot de eindsituatie te komen. Na deze verbetering is dit niet het aantal verplaatsingen van rolcontainers maar het aantal verplaatsingen van shuttles. Een shuttleverplaatsing is een verplaatsing waarbij de shuttle van pijpenbaan wisselt of waarbij de shuttle naar het andere uiteinde van de pijpenbaan rijdt.

Hieronder is beschreven hoe het aantal shuttleverplaatsingen berekend wordt. Bij deze berekening is het belangrijk om te weten dat de verplaatsingen van de rolcontainers uitgevoerd worden volgens het first in, first out principe. Daarnaast is het belangrijk om te weten dat de shuttles die er zijn aan een vast aantal pijpenbanen zijn toegewezen. In dit voorbeeld zijn er vier shuttles waarbij de toewijzing aan de pijpenbanen als volgt is:

- Shuttle 1: Pijpenbaan K01 tot en met pijpenbaan K04
- Shuttle 2: Pijpenbaan K05 tot en met pijpenbaan K08
- Shuttle 3: Pijpenbaan K09 tot en met pijpenbaan K11
- Shuttle 4: Pijpenbaan K12 tot en met pijpenbaan K14

Naast vaste pijpenbanen heeft de shuttle ook een startlocatie. Dit is de locatie waar de shuttle bij de start van het proces staat. De startlocaties in dit voorbeeld zijn:

- Shuttle 1: K01.T11
- Shuttle 2: K05.T11
- Shuttle 3: K09.T11
- Shuttle 4: K12.T11

De locatie K01.T11 betekent dat de shuttle zich onder pijpenbaan K01 en tegen transportbaan T11 bevindt. Aan de andere kant van de pijpenbaan ligt de transportbaan T12. Bij dit voorbeeld wordt er gebruikt gemaakt van een vijftal rolcontainerverplaatsingen die achtereenvolgens uitgevoerd worden. De verplaatsingen die bij het voorbeeld horen zijn:

- K04 naar K07 (T12)
- K05 naar K06 (T11)
- K10 naar K05 (T12)
- K14 naar K06 (T11)
- K10 naar K07 (T12)

Bij het berekenen van het aantal shuttleverplaatsingen is er verschil gemaakt tussen de volgende shuttleverplaatsingen:

- K01 (T11) naar K02 (T11)
- K01 (T11) naar K01 (T12)

Een shuttleverplaatsing naar een andere pijpenbaan wordt gezien als één shuttleverplaatsing. Een shuttleverplaatsing over dezelfde pijpenbaan van T11 naar T12 of andersom kost minder tijd en wordt gezien als een halve shuttleverplaatsing. Hieronder zijn de eerste en tweede rolcontainerverplaatsing uitgewerkt. Voor de overige drie verplaatsingen wordt alleen het aantal shuttleverplaatsingen weergegeven.

Bij de eerste opdracht zijn de shuttles 1 en 2 nodig. Deze staan geparkeerd op K01.T11 en K05.T11. Voor deze verplaatsing moet shuttle 1 van K01.T11 naar K04.T12 verplaatst worden. Dit wordt gezien als anderhalve shuttleverplaatsing. Dit omdat de shuttle eerst van K01.T11 naar K01.T12 verplaatst wordt. Dit is een halve verplaatsing. Hierna moet de shuttle van K01.T12 naar K04.T12 verplaatst worden. Dit is gelijk aan één shuttle verplaatsing. De tweede shuttle moet verplaatst worden naar K07.T12. Dit gebeurt vanuit K05.T11 en is ook gelijk aan anderhalve shuttleverplaatsing. Voor deze opdracht zijn er dus drie shuttleverplaatsingen nodig.

Voor de volgende opdracht is alleen de tweede shuttle nodig. Deze is na de eerste opdracht geparkeerd op K07.T12. Bij de tweede opdracht moet deze eerst naar K05.T11 verplaatst worden. Hierna moet de shuttle naar K06.T11 rijden. Deze opdracht maakt dus twee keer gebruik van deze shuttle. Het verplaatsen naar K05.T11 levert anderhalve shuttleverplaatsing op. De verplaatsing naar K06.T11 levert één shuttleverplaatsing op. Samen is dit twee en een halve shuttleverplaatsing.

Het aantal shuttleverplaatsingen voor de overige verplaatsingen zijn:

- K10 naar K05 (T12): drie shuttleverplaatsingen
- K14 naar K06 (T11): twee en een halve shuttleverplaatsing
- K10 naar K07 (T12): anderhalve shuttleverplaatsing

7.4.3 Minpunten per shuttleverplaatsing

Als laatste wordt het prioriteit cijfer aangepast. Hierin werd er na de eerste verbetering minpunten gegeven per uitgevoerde rolcontainerverplaatsing. De uitgevoerde rolcontainerverplaatsingen wordt hier vervangen door het aantal shuttleverplaatsingen. De tests 4 t/m 19 zijn uitgevoerd zodat bepaald kan worden wat het ideale aantal minpunten per shuttleverplaatsing is. De uitkomsten hiervan worden hieronder besproken. Ook wordt uitgelegd waarom er aan de eerste strategie geen minpunten toegekend worden en aan alle andere strategieën wel. De strategieën worden in de komende paragraaf weergegeven als: (Minpunten per shuttleverplaatsing; Vermenigvuldigingsfactor; Max punten positie rode rolcontainer).

Bij de eerste verbetering is besloten drie strategieën te gebruiken om het effect van de minpunten te testen. De drie strategieën zijn gekozen aan de hand van de bevindingen uit eerdere tests. In deze eerdere tests zijn de strategieën op volgorde gezet. Daarnaast is geconcludeerd dat strategieën met 0,975 of 0,99 als vermenigvuldigingsfactor in combinatie met minpunten weinig oplossingen genereren. Daarom wordt het effect van de minpunten niet op deze strategieën getest. De strategieën waar het effect van minpunten wel op getest wordt zijn:

- (0; 0,95; 50)
- (0; 0,95; 49)
- (0; 0,95; 55)

Het testen van de eerste strategie uit het rijtje is in deze paragraaf uitgewerkt. De andere twee strategieën zijn op dezelfde manier getest. Ze zijn echter niet in deze paragraaf beschreven. De conclusie die beschreven is geldt wel voor alle drie de geteste strategieën. De strategie is als volgt aangepast en getest:

- (0; 0,95; 50)
- (1,5; 0,95; 50)
- (1,4; 0,95; 50)
- (1,3; 0,95; 50)
- (1,2; 0,95; 50)
- (1,1; 0,95; 50)
- (1; 0,95; 50)
- (0,9; 0,95; 50)
- (0,8; 0,95; 50)
- (0,7; 0,95; 50)
- (0,6; 0,95; 50)
- (0,5; 0,95; 50)

Bij deze tests is het Branch and Bound principe uitgeschakeld. Dit zodat er onafhankelijk van elkaar bekeken kan worden welke van de strategieën de meeste oplossingen en de meest efficiënte oplossingen oplevert. De efficiënte van een oplossing is gelijk aan het aantal shuttleverplaatsingen. De efficiëntieafwijking wordt hier niet in meegenomen. Er zijn drie analyses gedaan:

- Aantal tests zonder oplossing voor de betreffende strategie
- Gemiddelde aantal shuttleverplaatsingen per strategie
- Aantal efficiëntste oplossingen per strategie

Aantal tests zonder oplossing voor de betreffende strategie:

Per strategie is bekeken hoeveel tests er geen oplossing hadden. In de tabel hieronder is voor alle strategieën weergegeven hoeveel tests er geen oplossing gegenereerd hebben.

Strategie	Aantal tests zonder oplossing
(0; 0,95; 50)	0
(1,5; 0,95; 50)	16
(1,4; 0,95; 50)	20
(1,3; 0,95; 50)	17
(1,2; 0,95; 50)	16
(1,1; 0,95; 50)	8
(1; 0,95; 50)	5
(0,9; 0,95; 50)	6
(0,8; 0,95; 50)	3
(0,7; 0,95; 50)	4
(0,6; 0,95; 50)	0
(0,5; 0,95; 50)	1

Figuur 7.4.3.1: Resultaten uitvoerbaarheidsanalyse

Uit de tabel is af te lezen dat er twee strategieën zijn die voor alle tests een oplossing hebben gegenereerd. Dit zijn de strategieën met 0 en 0,6 minpunten. Omdat de eerste eis het genereren van een oplossing is wordt één van deze twee strategieën als eerste uitgevoerd. Daarnaast is te zien dat een groter aantal minpunten resulteert in een groter aantal tests zonder oplossing.

Gemiddeld aantal shuttleverplaatsingen per strategie:

Ook is er naar het gemiddelde aantal shuttleverplaatsingen per strategie gekeken. In figuur 7.4.3.2 zijn de resultaten van deze analyse te zien.

Strategie	Gemiddeld aantal shuttleverplaatsingen
(0; 0,95; 50)	523,6734694
(1,5; 0,95; 50)	366,3428571
(1,4; 0,95; 50)	368,5
(1,3; 0,95; 50)	371,8970588
(1,2; 0,95; 50)	375,8979592
(1,1; 0,95; 50)	379,6511628
(1; 0,95; 50)	383,1666667
(0,9; 0,95; 50)	386,3723404
(0,8; 0,95; 50)	391,255814
(0,7; 0,95; 50)	401,5217391
(0,6; 0,95; 50)	402,9705882
(0,5; 0,95; 50)	410,6538462

Figuur 7.4.3.2: Resultaten efficiëntie-analyse

Met de gegevens uit de tabel kan geconcludeerd worden dat een groter aantal minpunten resulteert in een lager aantal shuttleverplaatsingen en dus een efficiëntere oplossing. Hierbij moet wel rekening gehouden worden met het feit dat in dit gemiddelde alleen tests met oplossing meegenomen zijn. In de vorige analyse was te zien dat een groter aantal minpunten resulteert in meer tests zonder oplossing.

Aantal efficiëntste oplossingen per strategie:

Ook is gekeken naar de individuele oplossingen per strategie in vergelijking met andere strategieën. De resultaten uit de analyse zijn hieronder weergegeven.

Strategie	Aantal beste oplossingen
(0; 0,95; 50)	0
(1,5; 0,95; 50)	12
(1,4; 0,95; 50)	10
(1,3; 0,95; 50)	7
(1,2; 0,95; 50)	7
(1,1; 0,95; 50)	7
(1; 0,95; 50)	8
(0,9; 0,95; 50)	4
(0,8; 0,95; 50)	3
(0,7; 0,95; 50)	1
(0,6; 0,95; 50)	0
(0,5; 0,95; 50)	0

Figuur 7.4.3.3: Resultaten individuele efficiëntie-analyse

Uit de tabel is af te lezen dat er drie strategieën zijn die geen enkele keer de efficiëntste oplossing gegenereerd hebben. Ook hier is een verband te zien tussen het aantal minpunten en het aantal efficiëntste oplossingen. Een hoger aantal minpunten zorgt voor meer efficiënte oplossingen.

Conclusie tests:

Met de resultaten uit de drie analyses wordt bepaald welke strategieën er gebruikt worden en in welke volgorde deze uitgevoerd worden. Allereerst zijn er drie strategieën die geen enkele keer de efficiëntste oplossing genereren. Deze kunnen daarom beter niet gebruikt worden. Omdat voor de strategieën die overblijven niet met zekerheid gezegd kan worden dat deze voor iedere situatie een oplossing genereren wordt één van deze strategieën alsnog uitgevoerd. Dit is de strategie waarbij er de grootste zekerheid is dat deze een oplossing genereert. Dit is:

- (0; 0,95; 50)

Bij de twee andere tests bleken er ook vijf strategieën te zijn die geen efficiëntste oplossing gegenereerd hadden. Omdat de strategie (0; 0,95; 50) ervoor zorgt dat er altijd een oplossing is worden alle andere strategieën zonder oplossing verwijdert.

Voor de overige strategieën wordt bepaald wat de juiste volgorde van uitvoeren moet zijn. Hierbij is de eerste in de rij de strategie die het beste werkt. Hierbij moet rekening gehouden worden met twee factoren:

- Het gemiddelde aantal verplaatsingen
- Het aantal tests zonder oplossing

Een hoger gemiddelde is niet per definitie beter omdat dit meer tests zonder oplossing oplevert. Daarom moet er een middelweg gekozen worden. Deze middenweg is bepaald door beide factoren samen te voegen volgens de volgende formule:

$$Factor = gemiddeld\ aantal\ stappen + aantal\ tests\ zonder\ oplossing * 2$$

Alle strategieën hebben een factorwaarde gekregen. Deze waarde is van laag naar hoog gezet. Hieronder zijn de strategieën weergegeven op de nieuwe volgorde.

1. (0; 0,95; 50)	16. (1,5; 0,95; 50)
2. (1,1; 0.95; 55)	17. (1,5; 0.95; 55)
3. (1,2; 0,95; 49)	18. (0,9; 0.95; 55)
4. (1; 0,95; 50)	19. (0,9; 0.95; 50)
5. (1; 0,95; 55)	20. (1,2; 0.95; 50)
6. (1,5; 0,95; 49)	21. (1,1; 0,95; 49)
7. (0,8; 0,95; 50)	22. (1; 0,95; 49)
8. (1,3; 0,95; 49)	23. (1,2; 0.95; 55)
9. (0,7; 0,95; 49)	24. (0,8; 0,95; 49)
10. (0,9; 0,95; 49)	25. (0,6; 0.95; 55)
11. (1,1; 0,95; 50)	26. (1,4; 0,95; 50)
12. (0,8; 0.95; 55)	27. (0,7; 0,95; 50)
13. (0,7; 0.95; 55)	28. (1,4; 0,95; 49)
14. (1,4; 0.95; 55)	29. (1,3; 0,95; 50)
15. (1,3; 0.95; 55)	

Figuur 7.4.3.4: De strategieën in volgorde van uitvoeren

7.4.4 Optimalisatie strategieën

Als optimalisatie is er gekozen de strategieën te testen met de negentien tests en in de volgorde zoals in ze figuur 7.4.3.4 staan. Uit de resultaten van de tests wordt bepaald of het nuttig is de volgorde te veranderen. Hierbij moeten de strategieën met de beste oplossingen als eerste uitgevoerd worden en dus vooraan staan. De resultaten uit de tests zijn hieronder weergegeven. Hierbij staat links het nummer van de test en rechts het aantal keer dat deze een betere oplossing gegenereerd heeft.

Nummer	Aantal oplossingen
0	49
1	49
2	27
3	9
4	13
5	16
6	4
7	7
8	3
9	2
10	7
11	0
12	3
13	4
14	2

Nummer	Aantal oplossingen
15	8
16	1
17	2
18	1
19	1
20	2
21	4
22	2
23	2
24	0
25	1
26	1
27	0
28	0

Figuur 7.4.4.1: Resultaten tests

Conclusie optimalisatie:

Uit de tests kan geconcludeerd worden wat de volgorde van de strategieën moet zijn zodat de beste oplossing zo snel mogelijk gevonden wordt. De strategieën worden in volgorde van nummer uitgevoerd. Door de strategieën een ander nummer te geven wordt de volgorde veranderd. In figuur 7.4.4.2 is te zien hoe de nummers veranderen.

Oud nummer	Nieuw nummer
0	0
1	1
2	2
3	5
4	4
5	3
6	9
7	7
8	12
9	14
10	8
11	25
12	13
13	10
14	15

Oud nummer	Nieuw nummer
15	6
16	20
17	16
18	21
19	22
20	17
21	11
22	18
23	19
24	26
25	23
26	24
27	27
28	28

Figuur 7.4.4.2: Nieuwe nummering

7.4.5 Conclusie derde verbetering

Omdat de graadmeter van de efficiëntie in deze verbetering veranderd is, kan er geen efficiëntieafwijking meer berekent worden. Dit betekent dat er geen vergelijking gemaakt kan worden tussen het verbeterde algoritme en de eerdere algoritmen. Daarom kan niet met zekerheid gezegd worden dat deze verbetering leidt tot een betere oplossing.

7.5 De overige verbeteringen

In de paragrafen 7.2 tot en met 7.4 zijn drie verbeteringen voor het algoritme uitgewerkt. In de inleiding van hoofdstuk zeven is gesproken over vijf verbeteringen. Door een te kort aan tijd zijn de laatste twee verbeteringen niet ingevoerd. De verbeteringen zijn:

- Het voorsorteren voor meerdere dagen
- Het voorsorteren voor meerdere dagdelen

Het invoeren van deze verbeteringen wordt als aanbeveling aan Logiqs B.V. gegeven.

8. Conclusie en aanbevelingen

In dit hoofdstuk worden de conclusies weergegeven en de aanbevelingen voor Logiqs B.V. gegeven. In paragraaf 8.1 wordt de conclusie per hoofdstuk beschreven. In paragraaf 8.2 staan de aanbevelingen voor Logiqs B.V. geschreven.

8.1 Conclusie

Probleemstelling:
Anthura Arndt heeft een probleem met het klaarzetten van de planten die de volgende dag verkocht worden. Het invoeren van de opdrachten die de planten naar de juiste plek moeten verplaatsen duurt vaak drie kwartier. Dit is veel te lang. Daarnaast staat de kas bij Anthura Arndt bijna vol met planten waardoor veel planten dubbel verplaatst moeten worden. Dit is inefficiënt. Een algoritme moet het probleem bij Anthura Arndt oplossen. Voor dit algoritme is er één eis gesteld: het algoritme moet voor alle startsituaties een oplossing genereren.

Analyse:
Er zijn twee analyses gedaan. Dit zijn de systeemanalyse en de tijdsanalyse. Uit de systeemanalyse is geconcludeerd dat de kas van Anthura Arndt bestaat uit veertien pijpenbanen, met op iedere pijpenbaan ruimte voor 42 rolcontainers. Op iedere pijpenbaan bevinden zich veertig rolcontainers wat betekent dat er 28 lege plaatsen in de kas aanwezig zijn. Ook is gebleken dat de rolcontainers de pijpenbanen aan twee kanten kunnen verlaten via een transportbaan. Een transportbaan bestaat uit machines die de rolcontainers kunnen verplaatsen. Verder is geconcludeerd dat Logiqs B.V. voor iedere machine en locatie in de kas een speciale naam heeft.

Als laatste is geconcludeerd dat de verplaatsing van een rolcontainer altijd bestaat uit de volgende instructies:

- Transportbaaninstructie: de transportbaan verplaatst de rolcontainer
- Shuttle-instructie (IN): een shuttle duwt de rolcontainer de pijpenbaan in
- Shuttle-instructie (UIT): een shuttle trekt de rolcontainer de pijpenbaan uit

Uit de tijdsanalyse is geconcludeerd dat een shuttle 66 seconden nodig heeft om een rolcontainer de pijpenbaan in te duwen. Deze verplaatsing wordt een shuttle-instructie(IN) genoemd. Omdat dit de instructie is die de meeste tijd nodig heeft is geconcludeerd dat de graadmeter voor efficiëntie gelijk wordt aan het aantal shuttle-instructies(IN). Omdat deze instructie in iedere verplaatsing éénmaal voorkomt is het aantal shuttle-instructie(IN) gelijk aan het totaal aantal rolcontainerverplaatsingen. Daarom is geconcludeerd dat de graadmeter voor efficiëntie gelijk is aan aantal rolcontainerverplaatsingen.

Literatuuronderzoek:

In de literatuur zijn vier methoden gevonden om het probleem van Anthura Arndt op te lossen. Dit zijn:

- De Breadth-first search
- De Depth-first search
- De Greedy search
- De Best-first search

De Breadth-first search en de Depth-first search gebruiken een methode waarmee alle opties nagegaan worden. Ook de opties die niet tot een situatie leiden die dichter bij de oplossing ligt. Om deze reden is geconcludeerd dat het bij beide methoden te lang duurt voordat deze tot een oplossing komen.

Bij de Greedy search en de Best-first search is dit anders. Bij deze methoden wordt beide gericht gezocht naar de beste uitgangssituatie. De uitgangssituatie wordt bepaald door een cijfer dat is opgebouwd uit meerdere factoren. Het verschil tussen beide is dat de Greedy search soms niet uitgewerkte situaties verwijdert terwijl de Best-first search deze bewaard. Omdat tussen deze verwijderde situaties ook goede uitgangssituaties zitten is geconcludeerd dat de Best-first search van deze vier de beste oplosmethode is.

Naast oplosmethodes zijn er in de literatuur ook toepassingen gevonden die het proces versnellen. De gevonden toepassingen:

- Het werken met meerdere lagen
- Het verwijderen van dubbele situaties
- Het beperken van opties

Uit de literatuur is gebleken dat het werken met meerdere lagen speciaal geldt voor de Greedy search. Omdat er voor de Best-first search gekozen is, is geconcludeerd dat het niet interessant is deze toepassing door te voeren. Over de overige twee toepassingen zegt de literatuur dat het de snelheid van de oplosmethode versneld. Om deze reden is geconcludeerd dat het interessant is deze toepassingen door te voeren.

Beginalgoritme:

Bij het programmeren van de Best-first search met de twee toepassingen is gebleken dat er een aantal bijzondere onderdelen ingebouwd moeten worden.

Bij het programmeren van de opties is geconcludeerd dat deze niet voor iedere situatie gelijk zijn. Een situatie in het beginstadium heeft andere opties nodig dan een situatie dit bijna in het eindstadium is. Daarom is besloten het proces op te delen in vier fases. In iedere fases zijn er andere opties mogelijk. De opties die er in een situatie zijn hangt daarom af van de fase waarin deze zich bevindt.

Na het programmeren van het algoritme is deze getest met 57 verschillende startsituaties. Er is allereerst getest of de tests allemaal een oplossing genereerde. Daarnaast is ook gekeken naar de gemiddelde tijdsduur en de gemiddelde efficiëntieafwijking van het algoritme. De efficiëntieafwijking is het verschil in verplaatsingen tussen de gevonden oplossing en de optimale oplossing uitgedrukt in percentages ten opzichte van het optimum. Uit de tests is gebleken dat er drie startsituaties waren die niet tot een oplossing kwamen. De gemiddelde tijdsduur van het algoritme was 73,86 seconden en de gemiddelde efficiëntieafwijking was gelijk aan 13,55%. Uit de tests is geconcludeerd dat het algoritme nog niet optimaal functioneert. Er is een tweede keer getest waarbij bekeken is waarom de drie tests niet tot een oplossing kwamen. Uit deze tests is geconcludeerd dat de opties en de factoren die de uitgangspositie bepalen niet optimaal waren.

Na het aanpassen van de opties en de factoren die de uitgangssituatie bepalen is er opnieuw getest. Bij deze test heeft het algoritme voor alle startsituaties een oplossing gegenereerd. Geconcludeerd kan worden dat de afstudeeropdracht hiermee voltooid is. Aan de enige eis die er was is immers voldaan. Daarnaast is uit de test naar voren gekomen dat het nieuwe algoritme sneller en efficiënter is dan het vorige algoritme. De gemiddelde tijdsduur is gedaald naar 59,52 seconden en de gemiddelde efficiëntieafwijking zakte naar 10,59%.

Wensen van Logiqs B.V.:

Nadat geconcludeerd is dat er aan de eis voldaan is heeft Logiqs B.V. een aantal wensen opgesteld die ze graag als verbetering wilde hebben. De wensen van Logiqs B.V.:

- De gemiddelde efficiëntieafwijking mag hoogstens tien procent zijn
- Het algoritme implementeren in de software van Logiqs B.V.
- De graadmeter voor efficiëntie aanpassen naar het aantal shuttleverplaatsingen
- Voorsorteren voor meerdere dagen
- Voorsorteren voor meerdere dagdelen

Het doorvoeren van de eerste wens is gedaan door meerdere strategieën achter elkaar te plaatsen. Hierbij is een strategie gedefinieerd als de combinatie van factoren die de uitgangspositie van een situatie bepalen. Door de factoren te veranderen ontstaan nieuwe strategieën. Deze nieuwe strategieën zijn achter elkaar geplaatst en getest. Uit de test bleek dat de gemiddelde efficiëntieafwijking van het algoritme met meerdere strategieën gelijk is aan 7,72%. Dit betekent een daling van bijna drie procent. Geconcludeerd kan worden dat het algoritme met meerdere strategieën voldoet aan de wens van Logiqs B.V.

De tweede wens is doorgevoerd door Koen van Leeuwen en Maarten van der Veen. Zij hebben de input en output van het algoritme gekoppeld aan de software van Logiqs B.V. Uit deze koppeling is gebleken dat de input en output van het algoritme juist gekozen is en goed aansluit op de software. Hierdoor is ook voldaan aan de tweede wens.

Bij de derde wens bleek dat er meerdere onderdelen veranderd moeten worden voordat deze wens functioneert. Het optimum dat nodig is voor het berekenen van de efficiëntieafwijking kan niet meer bepaald worden. Hierdoor kan geconcludeerd worden dat het algoritme na het toepassen van deze wens niet meer vergeleken kan worden met de eerdere algoritmen. Het is daarom niet duidelijk of deze wens echt een verbetering is.

Door een gebrek aan tijd zijn de laatste twee wensen van Logiqs B.V. niet doorgevoerd. Het doorvoeren van deze twee wensen wordt aan Logiqs B.V. aanbevolen.

Algemene conclusie:
Geconcludeerd kan worden dat het algoritme voldoet aan de eisen van Logiqs B.V. Het algoritme berekent automatisch welke verplaatsingen er nodig zijn om de planten op de juist plaats te krijgen. Daarmee is het handmatig invoeren van de opdrachten overbodig geworden. Dit scheelt Anthura Arndt heel veel tijd. Daarnaast is het ook gelukt het algoritme efficiënt te laten werken. Na een eerste efficiëntieafwijking van 13,55% is deze gedaald naar 7,72%. Dit is een flinke verbetering van bijna zes procent.

8.2 Aanbevelingen

In deze paragraaf zijn de aanbevelingen voor Logiqs B.V. beschreven. De eerste aanbevelingen die aan Logiqs B.V. gedaan worden zijn de wensen die Logiqs B.V. had maar die door een gebrek aan tijd niet doorgevoerd. Deze wensen zijn:

- Het voorsorteren voor meerdere dagen
- Het voorsorteren voor meerdere dagdelen

Beide wensen kunnen de toepassing in Dat-A-Logiqs aantrekkelijker maken om te verkopen. De eerste wens maakt het namelijk mogelijk om in het weekend alvast voor de hele week voor te sorteren. Hierdoor staan doordeweeks de meeste rolcontainers al voorgesorteerd. Alleen de rolcontainers die na het weekend de kas in gekomen zijn moeten dan nog verplaatst worden.

De tweede wens maakt het de tuinder mogelijk om meerdere verkoopmomenten op één dag te doen. Door voor te sorteren op dagdeel staan de planten die s ochtends verkocht worden vooraan. De planten die s middags verkocht worden staan hier achter.

Ook wordt het Logiqs B.V. aanbevolen om het algoritme universeel te maken. Op dit moment geldt het algoritme voor de kan van Anthura Arndt. Het algoritme kan ook gebruikt worden voor een kas met dezelfde rechthoekige indeling. Op een kas met een ingewikkeldere structuur werkt het algoritme niet. Door het algoritme universeel te maken is het algoritme op alle kasformaten te gebruiken.

Een andere aanbeveling is het aanpassen van het algoritme zodat het niet alleen op de logistieke systemen in de tuinbouw werkt maar ook op de systemen voor de palletopslag.

De laatste aanbeveling is het veranderen van de graadmeter voor de efficiëntie. Nu is ervan uit gegaan dat het efficiënt is om zo min mogelijk shuttleverplaatsingen te hebben. Door de graadmeter te veranderen naar de tijdsduur is de oplossing die de minste tijd nodig heeft om alles uit te voeren de oplossing die het beste is. Dit is een ingewikkelde graadmeter. Het is daarentegen wel de beste graadmeter die er is.

Literatuurlijst

Bouquiet, L. (2015, 4 maart). 4-3-2015 – Solving any Freecell game using Artificial Intelligence. https://www.youtube.com/watch?v=fI_dBJvAwe0

Vrije Universiteit Brussel. Hoofdstuk 9 Beslissingsalgoritmen. http://parallel.vub.ac.be/education/java/theorie/informatica%20II%20-%209%20-%20beslissingsalgoritmen.pdf

Bijlage I: Test beginalgoritme

In deze bijlage zijn alle testresultaten weergegeven. Per test zijn de uitkomsten in de tabel weergegeven. De tijd is weergegeven in seconden.

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
1	1	294	357	549	548			30,13	30,7			8413
1	1	294	357	549	548			32,6	33,08			8576
1	1	294	357	549	548			30,77	31,32			7793
1	2	294	521	573				40,13				6430
1	2	294	521	573				39,08				6737
1	2	294	521	573				36,85				7145
1*	3	294	523									4812
1*	3	294	523									5372
1*	3	294	523									5556
2	1	211	505	556	555	554	553	51,03	54,32	66,05	72,15	17200
2	1	211	505	556	555	554	553	49,96	53,68	58,48	70,8	17939
2	1	211	505	556	555	554	553	49,8	53,04	57,28	69,53	18525
2	3	211	504	572				97,5				13899
2	3	211	504	572				91,68				12047
2	3	211	504	572				113,49				9467
2*	2	211	503									7366
2*	2	211	503									8642
2*	2	211	503									8699
3	1	169	490	508				58,26				11277
3	1	169	490	508				56,89				11610
3	1	169	490	508				61,75				10087
3	2	169	474	546	545			66,61	66,83			5935
3	2	169	474	546	545			56,19	56,49			6047
3	2	169	474	546	545			70,63	70,82			5522
3	3	169	491	574				72,71				7716
3	3	169	491	574				69,40				7608
3	3	169	491	574				69,98				8805

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
4	1	100	416	477	476			85,07	109,86			10339
4	1	100	416	477	476			86,44	114,87			10048
4	1	100	416	477	476			86,64	115,41			9414
4	2	100	431	507	506			117,38	124,76			4475
4	2	100	431	507	506			107,69	114,63			4561
4	2	100	431	507	506			132,39	140,52			3409
4	3	100	405	474	473	472	471	94,43	101,79	111,43	113,75	6443
4	3	100	405	474	473	472	471	107,78	116,38	125,81	127,82	5861
4	3	100	405	474	473	472	471	98,98	108,39	113,37	114,29	6410
5	1	110	442	524				112,96				12758
5	1	110	442	524				104,67				13582
5	1	110	442	524				108,83				12858
5	2	110	428	507				100,08				5316
5	2	110	428	507				113,01				4742
5	2	110	428	507				94,85				5873
5	3	110	429	509				111,78				4063
5	3	110	429	509				121,73				3335
5	3	110	429	509				109,69				4051
6	1	120	442	525				91,95				9337
6	1	120	442	525				93,93				7309
6	1	120	442	525				101,41				7527
6	2	120	450	528				111,65				4582
6	2	120	450	528				107,27				4727
6	2	120	450	528				94,58				5667
6	3	120	453	546				106,94				4619
6	3	120	453	546				118,60				3914
6	3	120	453	546				122,03				4192
7	1	130	458	532	531	530		91,68	110	173,15		6497
7	1	130	458	532	531	530		92,77	106,19	168,34		6646
7	1	130	458	532	531	530		89,54	105,5	175,06		6292
7	3	130	458	528	527			80,09	95,08			6907
7	3	130	458	528	527			86,88	103,18			6064
7	3	130	458	528	527			90,95	104,67			6407
7*	2	130	451									2666
7*	2	130	451									2836
7*	2	130	451									3554

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
8	1	140	461	536	535			96,14	105,11			7542
8	1	140	461	536	535			95,54	107,21			6888
8	1	140	461	536	535			89,3	100			9027
8	2	140	449	510				81,42				7622
8	2	140	449	510				76,42				7360
8	2	140	449	510				88,42				6199
8	3	140	449	511	510			80,95	84,33			6882
8	3	140	449	511	510			74,97	78,51			7206
8	3	140	449	511	510			49,8	52,3			11086
9	1	150	465	534				89,58				4631
9	1	150	465	534				94,12				4273
9	1	150	465	534				103,42				3846
9	2	150	465	533				87,03				5004
9	2	150	465	533				82,09				5528
9	2	150	465	533				82,74				6327
9	3	150	473	525				76,89				5273
9	3	150	473	525				76,76				5154
9	3	150	473	525				78,11				5141
10	1	160	475	550	549			92,34	115,97			2952
10	1	160	475	550	549			93,17	115,07			3029
10	1	160	475	550	549			84,12	104,83			3391
10	2	160	454	498				70,89				4399
10	2	160	454	498				69,78				4244
10	2	160	454	498				71,15				4074
10	3	160	459	511				79,99				8361
10	3	160	459	511				84,92				6935
10	3	160	459	511				84,6				7563
11	1	170	471	527	525			79,58	80,15			6893
11	1	170	471	527	525			80,17	80,69			6936
11	1	170	471	527	525			82,16	82,68			6534
11	2	170	473	520				61,55				6858
11	2	170	473	520				60,1				6638
11	2	170	473	520				69,83				5977
11	3	170	474	533	531			53,56	56,17			10726
11	3	170	474	533	531			48,86	51,56			10667
11	3	170	474	533	531			48,56	51,47			11401

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
12	1	180	470	533	532			134,03	158,78			3635
12	1	180	470	533	532			135,69	160,03			3410
12	1	180	470	533	532			123,76	148,39			3825
12	2	180	476	555				49,21				7915
12	2	180	476	555				51,62				7344
12	2	180	476	555				52,08				6922
12	3	180	498	555				52,99				6376
12	3	180	498	555				58,15				5697
12	3	180	498	555				59,18				5629
13	1	190	482	530	529			72,57	72,87			7139
13	1	190	482	530	529			74,41	74,79			6616
13	1	190	482	530	529			70,5	70,8			6818
13	2	190	485	556	555			62,05	62,35			7839
13	2	190	485	556	555			59,96	60,32			7790
13	2	190	485	556	555			65,45	65,8			7521
13	3	190	486	541				66,81				6994
13	3	190	486	541				63,49				7081
13	3	190	486	541				71,98				5987
14	1	200	496	560				92,04				4725
14	1	200	496	560				87,58				5508
14	1	200	496	560				78,07				5824
14	2	200	512	547				58,6				14610
14	2	200	512	547				64,75				13096
14	2	200	512	547				56,62				13216
14	3	200	493	564				74,28				6538
14	3	200	493	564				69,57				6761
14	3	200	493	564				70,29				6637
15	1	210	504	547				77,56				2906
15	1	210	504	547				76,25				3157
15	1	210	504	547				75,59				3003
15	2	210	511	580				71,66				3264
15	2	210	511	580				65,89				3618
15	2	210	511	580				53,39				3745
15	3	210	497	581				62,67				4136
15	3	210	497	581				63,68				3961
15	3	210	497	581				60,09				3997

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
16	1	220	503	549	548			66,66	155,45			15629
16	1	220	503	549	548			63,55	139,68			18124
16	1	220	503	549	548			58,79	142,36			17296
16	2	220	501	587	586	585		58,42	76,13	85,35		10494
16	2	220	501	587	586	585		54,33	74,13	83,37		10516
16	2	220	501	587	586	585		61,37	80,48	90,14		10335
16	3	220	510	554				47,72				7981
16	3	220	510	554				44,59				8161
16	3	220	510	554				45,49				8250
17	1	230	509	558				50,78				10084
17	1	230	509	558				48,91				10228
17	1	230	509	558				47,73				10238
17	2	230	505	545				44,38				9053
17	2	230	505	545				42,56				9041
17	2	230	505	545				41,13				8172
17	3	230	509	565				47,45				11058
17	3	230	509	565				45,97				10399
17	3	230	509	565				42,64				10683
18	1	240	512	538	537	536	535	61,68	69,97	83,96	121,44	5630
18	1	240	512	538	537	536	535	73,02	77,36	93,12	133,53	4981
18	1	240	512	538	537	536	535	45,98	50,49	65,25	100,79	6655
18	2	240	510	578				50,72				9819
18	2	240	510	578				47,89				9717
18	2	240	510	578				49,32				10119
18	3	240	513	557				45,44				11751
18	3	240	513	557				40,35				10519
18	3	240	513	557				55,71				10253
19	1	250	515	577				73,46				2657
19	1	250	515	577				69,93				2629
19	1	250	515	577				61,43				2997
19	2	250	526	552	551			61,37	73,95			3358
19	2	250	526	552	551			55,06	66,46			3420
19	2	250	526	552	551			61,01	72,14			3246
19	3	250	517	562				58,09				3046
19	3	250	517	562				55,64				3222
19	3	250	517	562				64,64				2984

Bijlage II: Test verbeterd beginalgoritme

In deze bijlage zijn alle testresultaten weergegeven. Per test zijn de uitkomsten in de tabel weergegeven. De tijd is weergegeven in seconden.

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
1	1	294	357	482	481	480	479	36,48	36,62	36,74	36,82	7715
1	1	294	357	482	481	480	479	33,26	33,34	33,42	33,51	7671
1	1	294	357	482	481	480	479	33,28	33,36	33,45	33,53	7623
1	2	294	521	570	569	568	567	39,64	39,74	39,82	39,91	6897
1	2	294	521	570	569	568	567	39,01	39,11	39,21	39,3	6882
1	2	294	521	570	569	568	567	37,28	37,37	37,46	37,54	6912
1	3	294	523	590	589			45,32	47,96			6397
1	3	294	523	590	589			44,67	47,18			6394
1	3	294	523	590	589			40,17	43,94			6377
2	1	211	505	575	574			48,08	128,83			34218
2	1	211	505	575	574			44,03	127,19			34674
2	1	211	505	575	574			44,34	129,8			34013
2	3	211	504	569	568	567		43,18	50,05	103,62		31047
2	3	211	504	569	568	567		44,02	51,55	100,09		32029
2	3	211	504	569	568	567		41,13	48,39	99,3		32350
2	2	211	503	541				41,76				22188
2	2	211	503	541				38,89				22692
2	2	211	503	541				39,79				22475
3	1	169	490	509				53,63				18511
3	1	169	490	509				51,43				19171
3	1	169	490	509				52,72				19001
3	2	169	474	550				50,32				23035
3	2	169	474	550				51,49				22800
3	2	169	474	550				49,17				23219
3	3	169	491	545				50,84				25702
3	3	169	491	545				51,69				26037
3	3	169	491	545				49,63				25856

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
4	1	100	416	486				65,26				19049
4	1	100	416	486				67,38				18923
4	1	100	416	486				62,36				19538
4	2	100	431	494				75,3				14047
4	2	100	431	494				75,29				14319
4	2	100	431	494				77,32				13736
4	3	100	405	444				66,47				21288
4	3	100	405	444				66,27				23370
4	3	100	405	444				61,81				24072
5	1	110	442	503				79,55				13853
5	1	110	442	503				79,52				13887
5	1	110	442	503				81,66				13802
5	2	110	428	480				76,18				14085
5	2	110	428	480				78,74				13306
5	2	110	428	480				81,47				13747
5	3	110	429	476				83,14				14364
5	3	110	429	476				84,6				14490
5	3	110	429	476				84,31				14572
6	1	120	442	518				85,81				4979
6	1	120	442	518				87,38				4918
6	1	120	442	518				89,94				4879
6	2	120	450	509				87,64				5677
6	2	120	450	509				85,46				5642
6	2	120	450	509				88,07				5501
6	3	120	453	524				97,63				7809
6	3	120	453	524				99,5				7709
6	3	120	453	524				98,64				7755
7	1	130	458	503				73,18				21416
7	1	130	458	503				69,73				22072
7	1	130	458	503				71,67				21690
7	3	130	458	503				69,02				21982
7	3	130	458	503				71,64				21079
7	3	130	458	503				72,59				20993
7	2	130	451	496				62,84				30136
7	2	130	451	496				62,76				29877
7	2	130	451	496				63,61				29244

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
8	1	140	461	520				72,79				15883
8	1	140	461	520				69,28				16280
8	1	140	461	520				70,32				16109
8	2	140	449	489				58,03				24019
8	2	140	449	489				55,01				24688
8	2	140	449	489				56,31				24032
8	3	140	449	489				56,48				24431
8	3	140	449	489				56,72				24489
8	3	140	449	489				56,17				24725
9	1	150	465	511				75,37				13815
9	1	150	465	511				74,07				13856
9	1	150	465	511				76,98				13153
9	2	150	465	511				78,17				13506
9	2	150	465	511				74,7				13759
9	2	150	465	511				76,43				13443
9	3	150	473	505				70,15				11661
9	3	150	473	505				69,08				11510
9	3	150	473	505				71,92				11580
10	1	160	475	526	525			74,74	158,51			3557
10	1	160	475	526	525			77,51	157,42			3591
10	1	160	475	526	525			74,81	155,52			3629
10	2	160	454	511				69,63				4797
10	2	160	454	511				72,31				4818
10	2	160	454	511				70,55				4764
10	3	160	459	485				66,78				6096
10	3	160	459	485				66,63				6109
10	3	160	459	485				69,02				5915
11	1	170	471	536				49,49				23459
11	1	170	471	536				51,32				24224
11	1	170	471	536				48,17				24777
11	2	170	473	508				51,16				22513
11	2	170	473	508				50,8				22410
11	2	170	473	508				50,52				22176
11	3	170	474	521				49,41				28604
11	3	170	474	521				50,02				28689
11	3	170	474	521				50,25				29163

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
12	1	180	470	531				54,43				23309
12	1	180	470	531				53,41				23380
12	1	180	470	531				53,45				23486
12	2	180	476	536				51,86				29342
12	2	180	476	536				49,14				29566
12	2	180	476	536				49,37				29958
12	3	180	498	528				53,3				19932
12	3	180	498	528				54,05				20000
12	3	180	498	528				50,92				20470
13	1	190	482	528				59,85				12299
13	1	190	482	528				61,6				12184
13	1	190	482	528				59,49				12296
13	2	190	485	549				60,34				21333
13	2	190	485	549				58,6				20625
13	2	190	485	549				58,88				21178
13	3	190	486	527				52,4				23593
13	3	190	486	527				53,04				24033
13	3	190	486	527				49,7				24064
14	1	200	496	573	572	571		63,22	70,51	86,85		4249
14	1	200	496	573	572	571		62,31	69,99	86,7		4212
14	1	200	496	573	572	571		66,25	73,1	90,05		4058
14	2	200	512	563	562	561		71,4	77,9	78,58		4158
14	2	200	512	563	562	561		66,98	73,53	74,24		4328
14	2	200	512	563	562	561		67,55	74,58	75,27		4275
14	3	200	493	587				66				4412
14	3	200	493	587				64,7				4419
14	3	200	493	587				67,33				4232
15	1	210	504	563	562			62,35	68,15			4628
15	1	210	504	563	562			66,41	70,32			4693
15	1	210	504	563	562			62,98	67,05			4820
15	2	210	511	570	569			55,09	55,55			5699
15	2	210	511	570	569			61,58	62,28			4516
15	2	210	511	570	569			56,88	58,2			4558
15	3	210	497	563				64,17				5176
15	3	210	497	563				61,84				5241
15	3	210	497	563				66,14				5125

Test	Situatie	Aantal rode rolcontainers	Ondergrens	Aantal stappen tot:				Tijd tot:				Aantal doorgerekende situaties
				1ste oplossing	2de oplossing	3de oplossing	4de oplossing	1ste oplossing	2de oplossing	3de oplossing	4de oplossing	
16	1	220	503	583				55,9				30506
16	1	220	503	583				60,62				32013
16	1	220	503	583				57,56				32105
16	2	220	501	544	543			44,97	46,74			25467
16	2	220	501	544	543			42,56	43,66			25927
16	2	220	501	544	543			43,64	44,74			25885
16	3	220	510	533				41,35				23539
16	3	220	510	533				41,37				23430
16	3	220	510	533				41,59				23750
17	1	230	509	544	543			45,72	63,41			14911
17	1	230	509	544	543			44,4	62,54			15034
17	1	230	509	544	543			45,13	62,78			15026
17	2	230	505	527				44				16479
17	2	230	505	527				47,89				15895
17	2	230	505	527				49,88				15540
17	3	230	509	554				49				18219
17	3	230	509	554				46,27				19770
17	3	230	509	554				46,32				19761
18	1	240	512	527	526			46,8	47,97			8843
18	1	240	512	527	526			47,93	49,09			8922
18	1	240	512	527	526			48,31	49,51			8756
18	2	240	510	551				48,9				11565
18	2	240	510	551				51,48				11519
18	2	240	510	551				50,17				11614
18	3	240	513	543				50,72				12464
18	3	240	513	543				48,76				12554
18	3	240	513	543				47,85				12517
19	1	250	515	557	556	555		56,19	56,97	81,76		5055
19	1	250	515	557	556	555		54,05	54,85	79,44		5114
19	1	250	515	557	556	555		53,92	54,66	79,21		5196
19	2	250	526	549				49,63				5694
19	2	250	526	549				50,13				5608
19	2	250	526	549				49,76				5711
19	3	250	517	572	571			48,97	49,79			5721
19	3	250	517	572	571			49,67	50,47			5752
19	3	250	517	572	571			49,38	50,11			5728