Efficiënt inplannen van structurele transportorders

Master Project Business Analytics

Marloes Boxelaar

Stagebedrijf: PostNL Hoofdkantoor
Prinses Beatrixlaan 23
2595 AK Den Haag
Afdeling: Kwantitatieve Ondersteuning

Host begeleider: dr. Thije van Barneveld

Faculteit der Exacte Wetenschappen
De Boelelaan 1081a
1081 HV Amsterdam

Begeleider: dr. René Bekker
Tweede lezer: prof. dr. Ger Koole

Februari 2018

Voorwoord

De laatste uitdaging voor het afronden van de masterstudie Business Analytics aan de Vrije Universiteit van Amsterdam is een stageproject uitvoeren bij een bedrijf naar keuze. Dit afstudeerproject is uitgevoerd bij PostNL op de afdeling Kwantitatieve Ondersteuning in Den Haag, in de periode van juni 2017 tot en met januari 2018.

Ik zou graag mijn begeleider René Bekker aan de Vrije Universiteit willen bedanken voor al zijn hulp en advies gedurende mijn gehele stageperiode.

Mijn begeleider Thije van Barneveld van PostNL wil ik graag danken voor de wekelijkse begeleidingsgesprekken. Ik kon bij Thije terecht voor het stellen van vragen en hij dacht mee over oplossingen bij problemen waar ik tegen aan liep. Daarnaast wil ik Michiel Mueller van de afdeling Transport van PostNL in Hoofddorp graag dank zeggen voor het beantwoorden van al mijn vragen over het transportnetwerk van het bedrijf. Ook wil ik de collega's op de afdeling Kwantitatieve Ondersteuning bedanken voor de leuke periode die ik daar heb gehad tijdens mijn stage.

Natuurlijk wil ik ook mijn vriend Rob, familie en vrienden bedanken voor het motiveren gedurende deze stageperiode.

Marloes Boxelaar

Februari 2018

Abstract

De laatste jaren is de bezorgmarkt in Nederland van brieven-georiënteerd naar pakket-georiënteerd gegaan. Om als PostNL op de veranderende bezorgmarkt in te spelen is het van belang om na te gaan waar de effectiviteit in het transportproces verbeterd kan worden, rekening houdend met de eisen die PostNL stelt betreffende het behoud van de flexibiliteit en de hoge mate van betrouwbaarheid in het bezorgproces. De focus van deze scriptie ligt daarbij op de planningsfase van het groot transport van PostNL voor de structurele orders tussen de brieven- en pakketdepots.

Het doel van deze scriptie is om de ritten van de vrachtwagens tussen de brieven- en pakketdepots zo in te roosteren dat de totale ingeplande tijd geminimaliseerd wordt. De uitdaging is het feit dat het optimaliseren van het inroosteren van de orders een grote complexiteit (NP-moeilijk) met zich meebrengt. Naast dat er aan de tijdsvensters van het laden en lossen voldaan moet worden voor alle orders, moet een totale rit voldoen aan de arbeidstijden en voorwaarden van een chauffeur. Ook mag niet te lang gedaan worden over het genereren van een planning, aangezien de dagelijkse aangeleverde structurele orders binnen twee weken ingepland moeten worden op de bestaande planning.

Om de structurele orders te kunnen inroosteren zijn de volgende drie planningsmodellen gemaakt: *Integer Linear Programming (ILP)*, *Concurrent Scheduler Approach (CSA)* en *Genetic Algorithm (GA)*. Het ILP-model is hierbij een exacte aanpak, waarbij de overige twee methoden achtereenvolgens een heuristische en meta-heuristische aanpak betreffen. Hoewel het ILP-model geen oplossing vindt voor grote instanties van het probleem kan dit model een goede opzet geven om een planningsmodel op te bouwen en te testen voor een klein schaalmodel. Het CSA geeft PostNL binnen een redelijke rekentijd een (sub)optimale oplossing voor alle structurele orders. Het GA daarentegen loopt op tegen de ontoelaatbare oplossingen (ongeldige roosters) die tijdens het proces ontstaan doordat de tijdsvensters van het laden en lossen en arbeidstijden van een vrachtwagenchauffeur overschreden worden. Hierdoor vond het GA geen betere oplossing dan de aan het begin *random* gegenereerde oplossingen.

Als vervolgonderzoek kan gekeken worden naar het verbeteren van het *GA*, aangezien deze oplosmethode zeker potentie heeft om een goede oplossing te vinden. Bijvoorbeeld daar waar *mutation* plaatsvindt kan door sturing slechte (weinig aansluitende orders) roosters worden aangepast, zodanig dat er een toelaatbaar en goed rooster behouden blijft. Ook interessant is het om een gevoeligheidsanalyse te doen als de laad- en losvensters aangepast worden door het vergroten van het tijdsvenster met bijvoorbeeld tien minuten. Naast de tijdsrestrictie zou er ook onderzoek gedaan kunnen worden naar het scenario waarin de vrachtwagenchauffeur niet terug hoeft te keren naar het startdepot. Een ander vervolgonderzoek zou het maken van regio-afhankelijke planningen kunnen zijn, in plaats van het maken van een planningsmodel voor heel Nederland.

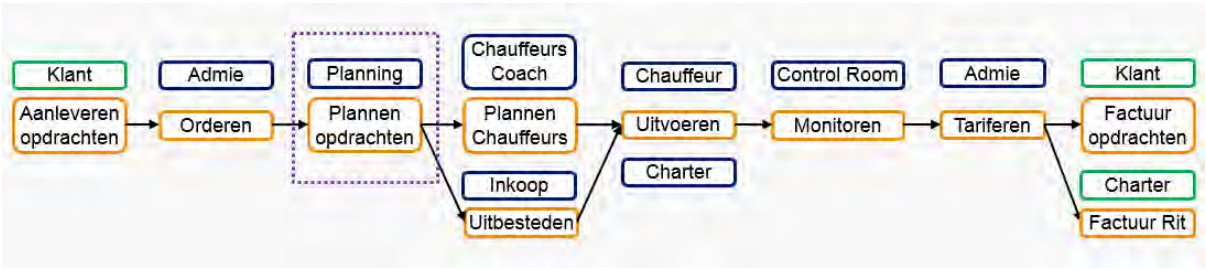
Inhoud

1 Inleiding

De afgelopen jaren is de digitale wereld niet meer weg te denken uit de Nederlandse maatschappij. Men bevindt zich steeds vaker in de wereld van het internet, onder andere op sociale media en webwinkels. Ook wordt deze wereld steeds toegankelijker door steeds beter en betaalbaar wordende mobiele apparaten. Deze digitalisering van de maatschappij heeft naast een groot economisch en maatschappelijke impact ook invloed op het bedrijfsleven en het gedrag van de consument. Hierbij wordt de consument steeds veeleisender door de transparantie van online informatie over prijs en kwaliteit van producten en diensten. Om als bedrijf toch stand te houden zal aan de toenemende wensen van de consument voldaan moeten worden door continu te innoveren.

Zo is de bezorgmarkt in Nederland van brieven-georiënteerd naar pakket-georiënteerd gegaan door het veranderende gedrag van de consument. Uit het onderzoek van de Autoriteit Consument & Markt (ACM)[1] bij de 28 deelnemende partijen op de postmarkt blijkt dat in 2016 het volume geadresseerde brievenbuspoststukken 10% lager was met 8% minder omzet in vergelijking met het voorgaande jaar. Daarentegen was in dezelfde periode het totale volume van het binnenlands pakketvervoer en het volume grensoverschrijdend pakketvervoer met 12% gestegen met 8% meer omzet bij 6 van de 18 deelnemende partijen die actief zijn in het vervoer van pakketten in Nederland. Hierbij is PostNL zowel de grootste partij in de brievensector als in de pakketsector [1].

Gezien pakketten in omvang meer ruimte in beslag nemen dan brieven, is er ook meer transportcapaciteit nodig om de producten van verzender naar ontvanger te brengen. Om daarbij de kwaliteit in de postmarkt te behouden is het belangrijk om het inplannen van het logistieke netwerk daarop af te stellen. Deze scriptie zal zich richten op de planningsfase van het transportproces van PostNL (zie Figuur 1).



Figuur 1: Transportproces van PostNL met als focus voor deze scriptie de planningsfase (gestippelde vierkant)

In het vervolg van dit hoofdstuk zal eerst het specifieke probleem voor deze scriptie gedefinieerd worden. Vervolgens zal de hoofdvraag met de deelvragen beschreven worden, met daarop aansluitend de structuurindeling van deze scriptie.

[1] De Autoriteit Consument & Markt (ACM) houdt zich sinds de liberalisering van de postmarkt in 2009 onder andere bezig met het toezicht op de ontwikkelingen op de postmarkt en sinds 2016 ook met de binnenlandse pakketmarkt.

1.1 Probleemdefiniëring

Om als PostNL op de veranderende postmarkt in te spelen is het van belang om na te gaan waar de efficiëntie in het transportproces verbeterd kan worden. Hierbij moet er rekening gehouden worden dat de visie van PostNL om zich te onderscheiden in klantgerichtheid door flexibiliteit en een hoge mate van betrouwbaarheid, behouden moet blijven.

Aangezien het bij PostNL een groot transportnetwerk betreft, is er voor deze scriptie gekozen om te richten op een gedeelte van dit netwerk. De focus zal zijn op de planningsfase voor het groot transport met ordersoort 60 en 70. Dit zijn de ritten in Nederland (interritten) die de vrachtwagens tussen de sorteercentra van brieven (ordersoort 60: INTER SCB) en pakketten (ordersoort 70: INTER SCP) afleggen. In het vervolg van deze scriptie zullen de sorteercentra als depots aangeduid worden, waarvan PostNL in totaal 6 brievendepots en 19 pakketdepots in Nederland heeft (zie Figuur 2; zie voor de adressen van deze depots Appendix C). Hierbij zijn de interritten opgedeeld in de volgende drie stromen tussen de depots:

- **Inter 1**: Afvoer vanuit depot naar depot+ (lossen bij een overslaglocatie)
- **Inter 2**: Afvoer vanuit depot+ naar depot (laden bij een overslaglocatie)
- **Inter 3**: Afvoer vanuit depot naar depot (zonder een overslaglocatie)

Hierbij is depot+ gedefinieerd als een depot of overslaglocatie voor de brieven en pakketten van en naar de andere depots. De inter 1 en inter 2 worden alleen gedaan als de vrachtwagen niet volledig gevuld kan worden met een rechtstreekse rit van het vertrekdepot naar het gewenste einddepot. In totaal heeft PostNL drie depot+ locaties die gevestigd zijn in Amersfoort, 's-Hertogenbosch en Waddinxveen.

Figuur 2: De huidige 25 PostNL brieven- en pakketdepots in Nederland voor het groot transport met ordersoort 60 en 70

Bij het aanvragen van een transportorder (lading vracht met brieven/pakketten afkomstig van een depot, dat bestemd is voor een ander depot) wordt er onderscheid gemaakt in drie typen: structureel, incidenteel en spoed. De kenmerken van deze typen orders zijn als volgt (zie Appendix B voor de uitgebreide kenmerk toelichting):

- **Structureel:** 3 Weken lang, 2 weken vooraf bekend
- **Incidenteel**: 1 Dag vooraf bekend
- **Spoed**: Dezelfde dag

Alle drie de ordertypes worden bij PostNL nog steeds op de oude methode (handmatig) ingepland en is daardoor behoorlijk tijdrovend en duur. Voor deze scriptie wordt voor de planning de focus gelegd op het type structurele orders. Onder het maken van een planning wordt het genereren van ritten (diensten bestaande uit orders) voor de vrachtwagens tussen de depots verstaan. Elke order heeft daarbij specifieke tijdsvensters om de vracht te kunnen laden en lossen, met daarnaast een procestijd. Hierbij is de procestijd gedefinieerd als de totale tijd die een vrachtwagen nodig heeft om de vracht van een order te kunnen laden, rijden en lossen. Naast dat alle orders binnen de tijdsvensters uitgevoerd moeten worden, geldt voor de ritten dat deze moeten voldoen aan de arbeidstijden en voorwaarden van een chauffeur. De bedoeling is om de orders zo in te roosteren dat een rit zoveel mogelijk gevuld is en dat er zo min mogelijk onbenutte tijd tussen komt te zitten. Dat wil zeggen: het voorkomen dat vrachtwagens leeg rijden naar hun vervolg depot, of moeten wachten tot er geladen of gelost kan worden (wachttijd). Uiteindelijk is het de bedoeling om de ritten van de vrachtwagens tussen de depots zo in te roosteren dat de totale ingeplande tijd geminimaliseerd wordt.

1.2 Onderzoeksvragen

Het doel van deze scriptie is om na te gaan of de ritten (diensten bestaande uit de structurele orders van PostNL met ordersoort 60 en 70) van de vrachtwagens efficiënter ingepland kunnen worden door gebruik te maken van wiskundige modellen. Om hieraan te voldoen is hieronder de hoofdvraag beschreven met daaropvolgend de deelvragen die helpen om deze onderzoeksvraag te kunnen beantwoorden.

Hoofdvraag:

"Hoeveel efficiënter kan PostNL haar planning voor de structurele orders maken door gebruik te maken van wiskundige modellen?"

Deelvragen:

- *Hoe zijn de structurele orders over de depots verdeeld?*
- *Welke modellen om het beoogd probleem van PostNL op te lossen zijn bekend in de literatuur?*
- *In hoeverre is het probleem exact op te lossen met een Integer Linear Programming (ILP) model?*
- *Kan een heuristiek of meta-heuristiek op basis van het eerder geformuleerde ILP-model en het literatuuronderzoek een oplossing geven binnen een redelijke rekentijd?*
- *Indien de nieuwe oplossing een lagere totale ingeplande tijd heeft, welke vrachtwagens moeten er afgeschaald/opgeschaald worden t.o.v. de vorige planning?*

De uitdaging voor deze scriptie zit hem in het feit dat het optimaliseren van het inroosteren van de orders een grote complexiteit met zich meebrengt. Naast het voldoen aan de tijdsvensters van het laden en lossen voor alle orders, moet een totale rit ook voldoen aan de arbeidstijden en voorwaarden van een chauffeur. Daarnaast mag niet te lang gedaan worden over het genereren van een planning, aangezien de dagelijkse aangeleverde structurele orders binnen twee weken ingepland moeten worden op de bestaande planning.

1.3 Structuur van de scriptie

Deze scriptie is opgebouwd uit zeven hoofdstukken en vijf bijlagen. Na de inleiding volgt hoofdstuk 2 met een literatuuronderzoek over classificatie van het soort variant van het *Vehicle Routing Problem (VRP)* waar het transportprobleem van PostNL onder valt. Verder zal in dit hoofdstuk uitleg gegeven worden welke oplosmethoden uit de literatuur bekend zijn, met daaropvolgend de redenering voor de gebruikte oplosmethoden in deze scriptie. In hoofdstuk 3 zal eerst de ontvangen PostNL-data worden beschreven, daarna volgt een data-onderzoek met de gemaakte aannames. In hoofdstuk 4 begint de formulering en implementatie van de exacte oplosmethode. In hoofdstuk 5 wordt de gebruikte heuristische en meta-heuristische oplosmethode behandeld. Vervolgens wordt in hoofdstuk 6 de resultaten van de drie gebruikte oplosmethoden beschreven. Het laatste hoofdstuk geeft de conclusies voor de gevonden resultaten, de restricties en de aanbevelingen voor vervolgonderzoek.

2 Literatuuronderzoek

Een van de bekendste problemen op het terrein van de combinatorische optimalisatie is wel het handelsreizigersprobleem (*Travelling Salesman Problem, TSP)*. Veel onderzoek is gedaan naar verschillende varianten op het TSP. Deze scriptie richt zich op de gegeneraliseerde versie van het TSP, genaamd het *Vehicle Routing Problem (VRP)*. Het VRP is in 1959 geïntroduceerd door Dantzig & Ramser [3] met als doel om een optimale route (volgorde van bezoeken aan klanten) te vinden, waarbij het aantal vrachtwagens en de totale afgelegde afstand geminimaliseerd moeten worden. Daarnaast moet gelden dat elke vrachtwagen zijn route start en eindigt bij het hoofddepot en dat de capaciteit van een vrachtwagen niet overschreden mag worden door de meegenomen vracht voor de klanten. Om het VRP ook te kunnen toepassen in het dagelijks logistieke netwerk zijn in de literatuur verscheidende onderzoeken uitgevoerd op specifieke situaties die in de praktijk ook voorkomen. Er kan gedacht worden aan het toevoegen van meerdere depots, vrachtwagens die niet terug hoeven te rijden naar het hoofddepot of vracht aanleveren aan de klant binnen bepaalde tijdsvensters.

Het probleem van deze scriptie valt het best te categoriseren onder de variant *Multi-Depot Pickup and Delivery Problem with Time Windows (MDPDPTW),* wat een generalisering is van het *Vehicle Routing Problem with Multiple-Depots with Time Windows (MDVRPTW)*. Bij beiden is er sprake van meerdere depots en tijdsvensters voor de te leveren vracht bij de klant. Alleen bij het MDVRPTW vertrekken vrachtwagens vanuit hun toegewezen hoofddepot met de benodigde vracht voor de klanten, terwijl bij het MDPDPTW vertrekt de vrachtwagen leeg van het hoofddepot en dient de klant als laad- of loslocatie voor de vracht van andere klanten. Vergeleken met het MDPDPTW wijkt het beoogd probleem van PostNL gedeeltelijk hiervan af, omdat het transport van de PostNL orders zich alleen tussen de depots afspeelt. Voor een vrachtwagen dient een van deze depots als zijn hoofddepot. Hierdoor kunnen orders die een vrachtwagenchauffeur uitvoert ook van of naar zijn toegewezen hoofddepot plaatsvinden. Daarnaast wordt een vrachtorder altijd volledig geladen of gelost.

Aangezien het MDPDPTW onder de complexiteitsgraad van NP-moeilijk valt [11], wil zeggen dat er geen algoritme is om het probleem in polynomiale tijd op te lossen naarmate het probleem groter in omvang wordt. Om na te gaan welke oplosmethoden geschikt kunnen zijn voor het MDPDPTW is er gebruik gemaakt van het uitgebreide literatuuronderzoek van Montoya-Torres e.a. [13] over verschillende varianten van het *Vehicle Routing Problem with Multiple-Depots* (*MDVRP)*. Hierin worden de onderzochte *papers* onderverdeeld in de volgende drie oplosmethoden: exact, heuristisch en meta-heuristisch. Hierbij kunnen de *papers* die een exacte oplosmethode hanteren meestal alleen een optimale oplossing vinden voor kleine instanties van het probleem. Dat wil zeggen: problemen met een zeer gelimiteerd aantal orders, klanten of locaties. Bij het toepassen van een exacte methode voor *real-world* situaties is het aantal variabelen hoog, waardoor het oplossen van het probleem al snel te gecompliceerd wordt [17]. Voor het vinden van een oplossing voor grotere probleem instanties wordt er vaak naar een heuristiek uitgeweken. Met het gebruik van een heuristiek (systematische aanpak) is er geen garantie op het vinden van een optimale oplossing, maar wel dat er een goede oplossing gevonden kan worden binnen een redelijke rekentijd [6]. Nadeel van heuristieken is wel dat er een grote kans bestaat om vast te komen zitten in een lokaal optimum door het relatief kleine zoekgebied van oplossingen. Om uit het lokaal optimum te kunnen ontsnappen wordt er vaak voor een meta-heuristische aanpak gekozen [6]. Dit is een probleem-onafhankelijke techniek die soms ook

slechte oplossingen tijdelijk accepteert, waardoor het mogelijk wordt gemaakt om ook buiten het oplosgebied te zoeken wat de kans op een betere oplossing verhoogd.

In de literatuur is het aantal *papers* over het MDPDPTW erg schaars. Uit het uitgebreide literatuuronderzoek van Montoya-Torres e.a. [13] worden slechts vier *papers* onder het type MDPDPTW met één enkele doelstelling, zoals bijvoorbeeld het minimaliseren van het totaal aantal ingezette vrachtwagens, gecategoriseerd (zie Tabel 1).

		Oplossingsmethode		
Jaar	Referentie	Exact	Heuristisch	Meta-heuristisch
2007	Pisinger & Ropke [18]			•
2008	Goela & Gruhn [7]		•	
2009	Flisberg e.a. [6]	•		•
2010	Sombunthama & Kachitvichyanukulb [19]		•	

Tabel 1: Literatuuronderzoek met papers die vallen onder de categorie MDPDPTW met één enkele doelstelling, van [13] aangepast voor de doelstelling van deze scriptie

In de paper van Pisinger & Ropke (2007) worden vijf verschillende varianten van het VRP getransformeerd tot het *Rich Pickup and Delivery Problem with Time Windows (RPDPTW)*. Hierbij heeft elke order een laad- en loslocatie. Het doel is om alle orders over de beschikbare vrachtwagens te verdelen, waarbij de totale afgelegde afstand geminimaliseerd moet worden. In de genoemde paper wordt een iteratief proces toegepast op het RPDPTW, genaamd het *Adaptive Large Neighborhood Search (ALNS)*. Hierbij wordt in elke iteratiestap een gedeelte van de huidige oplossing verwijderd en opnieuw samengesteld in de hoop dat het naar een betere oplossing leidt. Het nadeel van het RPDPTW is dat het alleen meerdere hoofddepots ondersteunt - bij de transformatie van MDVRP tot RPDPTW - als elke order vooraf aan een specifiek hoofddepot wordt toegewezen. Dit zou voor PostNL betekenen dat een order alleen door vrachtwagens van zijn toegewezen hoofddepot uitgevoerd kunnen worden en niet door de vrachtwagens van de andere depots. Aangezien bij het gestelde probleem van PostNL het onbekend is welke order tot welk hoofddepot behoort, zou de toewijzing van de orders aan een vast hoofddepot het zoeken naar een oplossing in het oplosgebied beperken.

Goela & Gruhn (2008) beschrijven een combinatie van het gegeneraliseerde VRP en het *Pickup and Delivery Problem (PDP)* voor verscheidende *real-life* toepassingen (bijvoorbeeld het gebruik van tijdsvensters of verschillende start- en eindlocaties voor de wagens). Enig criterium is wel dat het aantal vrachtwagens voor elke hoofddepot vooraf vastgezet moet worden, omdat het een dynamische planning betreft. Dit zou een beperking zijn voor het probleem van PostNL, aangezien de vrachtwagens tussen elk van de depots kunnen rijden en het daardoor onbekend is hoeveel vrachtwagens er per hoofddepot toegewezen moeten worden.

Het doel in de paper van Flisberg e.a. (2009) is om voor vrachtwagens van een Zweeds houtopslagbedrijf een routeplanning te maken. De beschreven oplosmethode maakt gebruik van een twee-fasen aanpak, waarbij het probleem wordt getransformeerd tot een standaard VRPTW. Enig nadeel is dat de planningsmethode geen rekening houdt met eventuele wachtrijen die kunnen ontstaan bij de laad- en loslocaties. Wat bij het beoogd probleem voor PostNL wel het geval kan zijn als er nog geen ruimte of personeel is om de vracht te laden of te lossen.

In Sombunthama & Kachitvichyanukulb (2010) wordt het probleem beschreven als het *Generalized Vehicle Routing Problem for Multi-Depot with Pickup and Delivery Requests (GVRP-*

MDPRDR). Wel moet gezegd worden dat deze *paper* verkeerd gecategoriseerd is in de overzichtspaper van [13], omdat het zelfs drie doelstellingen in plaats van één heeft en er geen heuristiek, maar een meta-heuristiek als oplosmethode gebruikt is. Het doel van Sombunthama & Kachitvichyanukulb is om zowel het aantal wagens als de totale afstand te minimaliseren, terwijl gelijktijdig het aantal orders gemaximaliseerd wordt. Voor het oplossen van het probleem wordt er gebruikt gemaakt van een *Particle Swarm Optimization (PSO)* algoritme met meerdere sociale leerstructuren. Het nadeel van het PSO algoritme is dat hierbij het hoofddepot niet als laad- of loslocatie mag worden gebruikt, terwijl dat wel als een vereiste wordt gezien voor het transportprobleem van PostNL.

De beschreven *papers* over het MDPDPTW voldoen niet volledig aan het beoogd transportprobleem van PostNL. Daarnaast is het omzetten van de restricties erg lastig omdat een groot gedeelte van het model daarvoor moet worden aangepast. Daarom is er voor deze scriptie gekozen om het model voor PostNL op te bouwen uit de beter gedocumenteerde variant *Vehicle Routing Problem with Time Windows (VRPTW)*. Om inzicht te krijgen in welke restricties nodig zijn en om na te gaan waar de grens ligt om het model van PostNL exact op te lossen, is er gekozen om een *Integer Linear Programming (ILP)* te implementeren. ILP is een methode waarbij de optimale waarde van een lineaire functie over meerdere beslissingsvariabelen wordt bepaald. Een van de grondleggers van lineaire programmering is George Dantzig, die in 1947 de simplexmethode heeft uitgevonden om een lineair optimalisatieprobleem op te kunnen lossen, of de onoplosbaarheid ervan vast te kunnen stellen. De implementatie voor deze scriptie zal gebaseerd zijn op de wiskundige formulering gebruikt in Van Barneveld [21] en Neumann [14] voor het inroosteren van opdrachten.

Aangezien er verwacht wordt dat PostNL te veel transportorders heeft om het model exact op te lossen en het inroosteren tijdsgebonden is, is er gezocht naar een *greedy* heuristiek. In [9] wordt zo een heuristiek genaamd *Concurrent Scheduler Approach (CSA)* gebruikt voor het *Vehicle Routing Problem with Time-Windows* die eenvoudig en snel een toelaatbare oplossing kan vinden. Gezien het feit dat dit een VRPTW betreft is het voor het transportmodel van PostNL nodig om dit om te zetten naar een model waar meerdere hoofddepots ingezet kunnen worden en dat de verzameling van klanten anders geformuleerd wordt. Met dit laatste wordt bedoeld dat een locatie van een klant, de locatie van een PostNL depot betreft die door vrachtwagens meerdere keren bezocht kan worden. Deze locatie heeft daarmee een laad- of losplaats voor vracht en is tevens hoofddepot voor de vrachtwagens die daar hun route beginnen of eindigen.

Om te controleren in hoeverre er een betere oplossing gevonden kan worden voor het gemaakte heuristische model, wordt in deze scriptie ook een meta-heuristische aanpak gedaan. In de literatuur wordt veelal voor het oplossen van varianten van VRP problemen een G*enetic Algorithm (GA)* gebruikt [18]. In [15] wordt een GA toegepast dat zelfs betere oplossingen kan vinden voor vier instanties uit het bekend benchmark probleem van Augerat en voor twee instanties uit het vaak toepaste benchmark probleem van Christofides & Eilon [2]. De verklaring voor deze goede resultaten voor het toegepaste GA is door het gebruik van een nieuwe representatie van een rooster. Ook heeft deze representatie potentie om uitgebreid te worden naar een VRP met tijdsvensters. Daarom is er voor deze scriptie gekozen om als meta-heuristische aanpak het GA te nemen gebaseerd op [16]. Het beoogd model voor PostNL zal hierbij stapsgewijs opgebouwd en onderbouwd worden met de beschreven theorie over evolutionaire algoritmen uit [6].

3 Data

In dit hoofdstuk zal eerst de ontvangen data beschreven worden. Daarna volgt er een onderzoek naar de verdeling van het werkpakket van de transportorders gedurende de betreffende periode. Aan het eind worden de genomen aannames beschreven die volgen uit de databeschrijving en het data-onderzoek.

3.1 Databeschrijving

De ontvangen data van PostNL bestaat uit alle transportorders van zondag 16 juli tot en met zaterdag 22 juli 2017 (werkweek 29) die zijn uitgevoerd. Een order kan beschreven worden als een laad- en losorder die uitgevoerd moet worden door een vrachtwagen binnen de gegeven tijdsvensters. In totaal bevat de dataset 17.429 orders, waarvan elke order 37 variabelen heeft. De dataset bevat naast de drie verschillende ordertypen (structureel, incidenteel en spoed) ook 35 verschillende ordersoorten. In deze scriptie wordt de focus gelegd op het type structurele order met ordersoort 60 (INTER SCB: interritten voor sorteercenter brieven) en 70 (INTER SCP: interritten voor sorteercenter pakketten). Dit resulteert in een totaal van 3.837 orders gedurende de week.

Attributen	Beschrijving	Denotatie
Ordernr	Het nummer van de order	“2541627”, “2541628”
Opdrachtdatum	Datum wanneer de order start	<dd-mm-jjjj>
Laadcode	Locatie voor het laden: postcode + huisnummer + eerste letter depotnaam	“1046BP100S”, “3439NT80S”
LaadtijdBegin	Vroegst mogelijke begintijd voor het laden van de order	<uu:mm>
LaadtijdEind	Laatst mogelijke eindtijd voor het volledig volladen van de order	<uu:mm>
Loscode	Locatie voor het lossen: postcode + huisnummer + eerste letter depotnaam	“3439NT80S”, “3062CZ300S”
LostijdBegin	Vroegst mogelijke begintijd voor het lossen van de order. Gelijk aan “LaadtijdBegin”	<uu:mm>
LostijdEind	Laatst mogelijke eindtijd voor het volledig lossen van de order	<uu:mm>
SoortOrder	Soort van de order in nummers	“60” of “70”

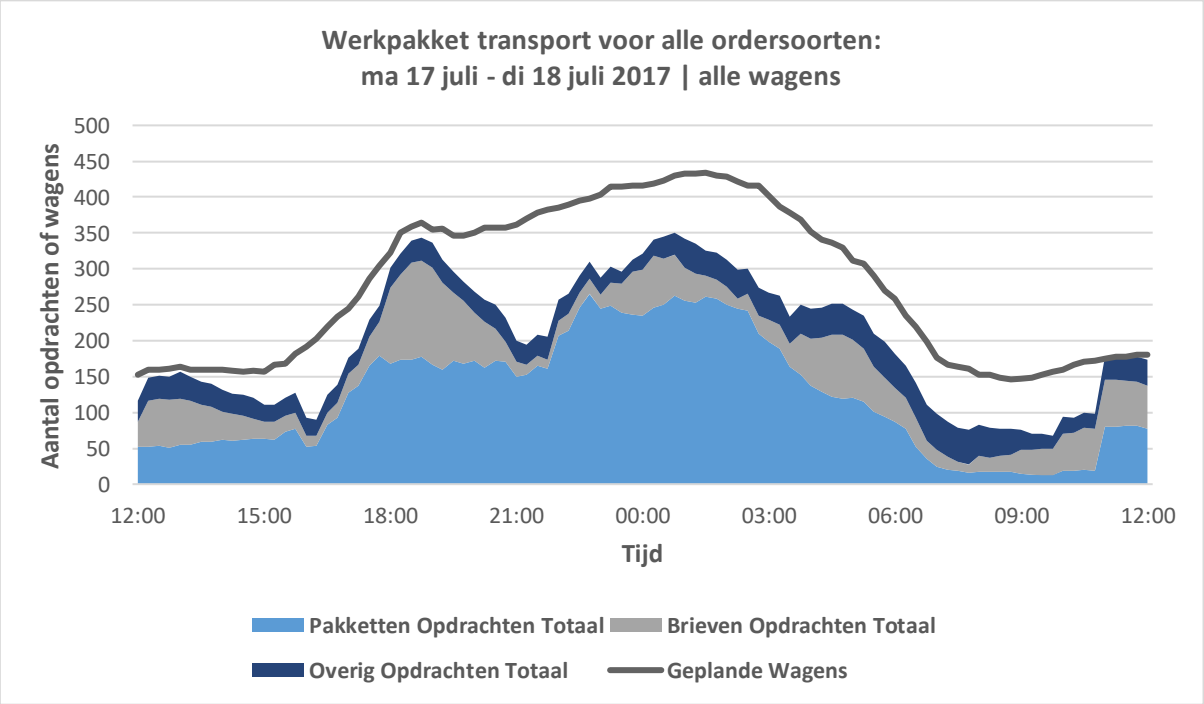
Tabel 2: Beschrijving en denotatie van de geselecteerde attributen uit de ontvangen orderdata van werkweek 29 in 2017

Als maat voor de efficiëntie van een planning is er gekozen voor de totale ingeroosterde tijd van alle ingezette vrachtwagens. Hiervoor is het eerst belangrijk om te weten welke elementen een planning vormen. In de transportplanning wordt er per vrachtwagenchauffeur een route gemaakt die start en eindigt bij hetzelfde depot, met daartussenin orders met hun betreffende laad- en losplaatsen en geplande aankomst- en vertrektijden. Eventuele wachttijden zijn tussen de orders in de planning verwerkt. Om een planning te kunnen genereren zijn de volgende gegevens nodig: ordernummers, de laad- en loslocatie, tijdsvensters voor het laden en lossen, en de procestijden van de orders. De procestijd van een order kan beschreven worden als de totale tijd die nodig is om het gehele proces van het laden, rijden van de laadplaats naar de losplaats, en het lossen te kunnen voltooien. Binnen het laad/losproces valt de rijtijd op het terrein, het manoeuvreren, het laden/lossen, het afhandelen van vrachtbrieven en het verzegelen tot het moment waarop de auto de openbare weg oprijdt. Zowel ordersoort 60 als 70 hebben daarin gelijke constante laad- en lostijden van 25 minuten. Op de attribuut

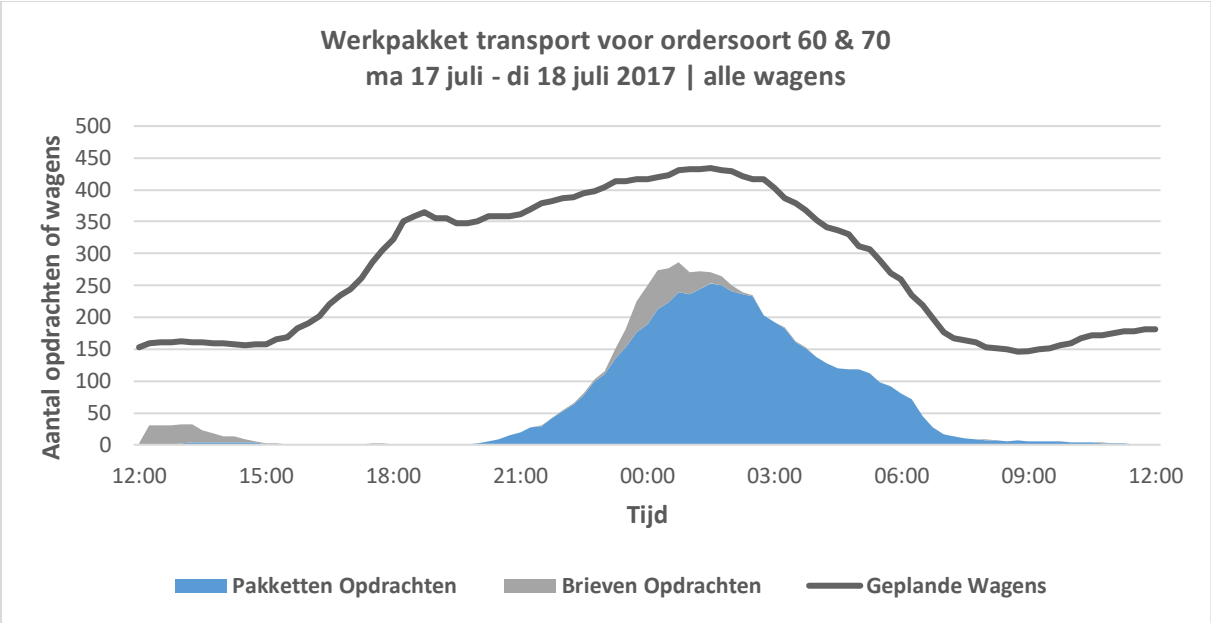
procestijd na zijn alle benodigde gegevens uit de orderdata gehaald (zie Tabel 2 voor de beschrijving en denotatie van deze attributen, en Appendix D voor een datavoorbeeld).

3.2 Data-onderzoek

Voor het inzichtelijk maken van de data is er eerst gekeken naar het verloop van alle soorten transportorders bij elkaar gedurende de werkweek. Uit de data blijkt per dagdeel een patroon waarneembaar te zijn dat in het begin van de avond en rond middernacht de hoogste pieken voor het aantal opdrachten laat zien. In Figuur 3a is dit patroon te zien voor de maandagmiddag tot en met de dinsdagmiddag voor de betreffende week. Verder ligt het aantal orders van pakketten gedurende de dag aanzienlijk hoger dan die van de categorie brieven en overig. Naast de orders is de variabele “geplande wagens” ook in het genoemde Figuur 3a opgenomen. Hierbij is deze variabele gelijk aan de variabele “aantal opdrachten”, aangezien er voor elke order altijd één wagen nodig is. De variabele “geplande wagens” ligt gedurende de nacht het hoogst aangezien dan de meeste orders uitgevoerd worden. Om te kunnen vaststellen in welk tijdsinterval de structurele orders met ordertype 60 en 70 plaatvinden is uit diezelfde dataset Figuur 3b gemaakt. Hierin is te zien dat er een kleine piek waarneembaar is voor het aantal orders voor brieven gedurende maandagmiddag en dinsdagnacht, terwijl het grootste gedeelte van de pakketopdrachten in het tijdsvenster van 20:00 - 8:00 uur plaatsvindt.

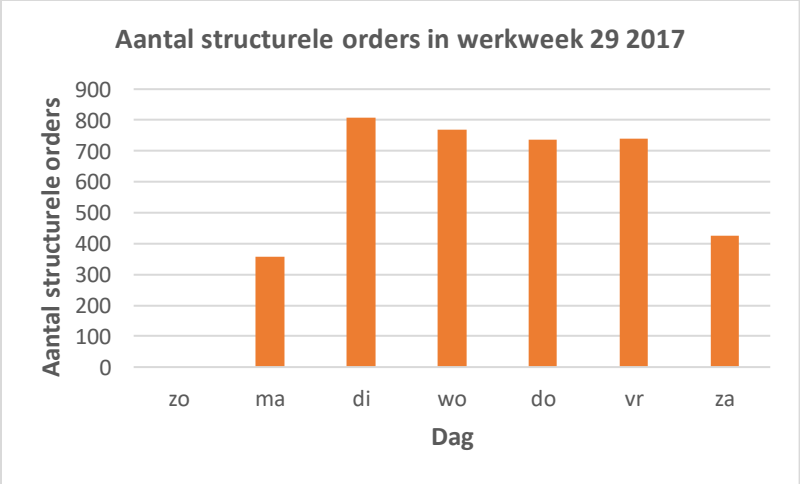


Figuur 1a: Het totale werkpakket transport van maandagmiddag tot dinsdagmiddag in werkweek 29 2017



Figuur 3b: Het werkpakket transport voor ordersoort 60 en 70 van maandagmiddag tot dinsdagmiddag in werkweek 29 2017

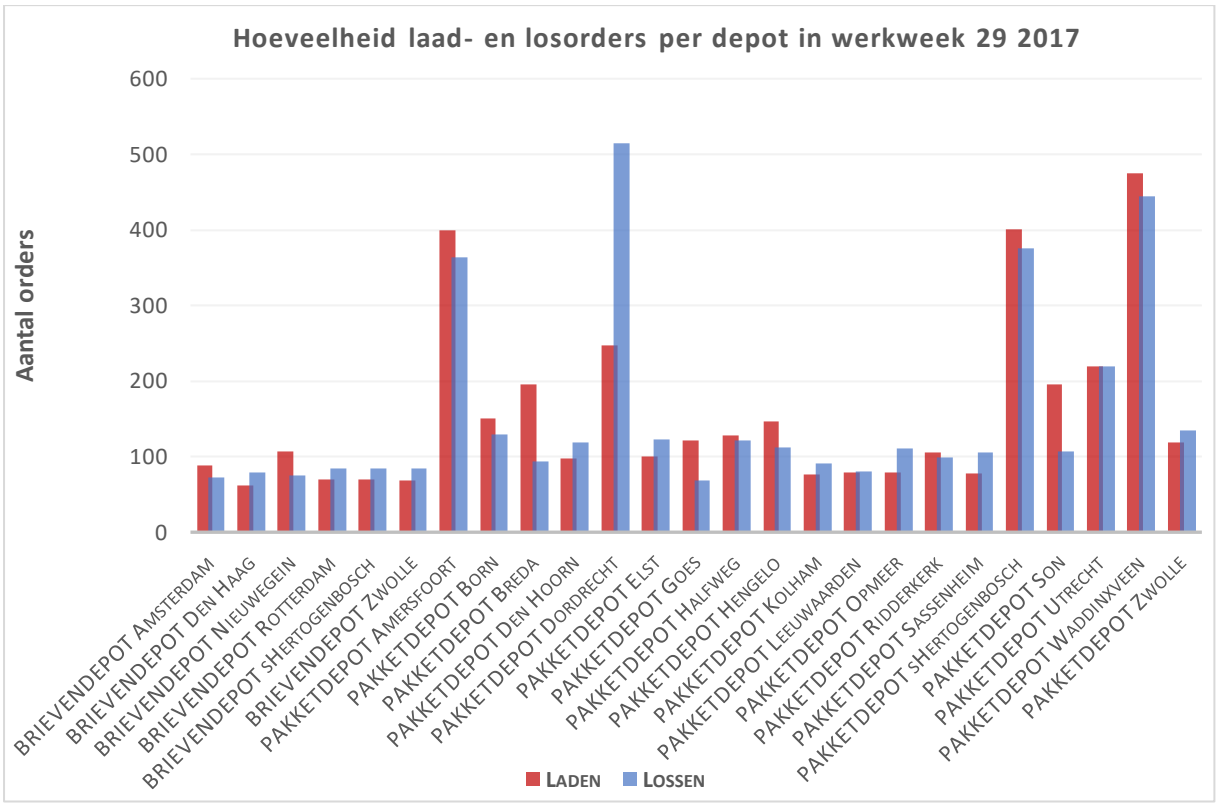
In Figuur 4 is het aantal structurele orders voor de betreffende ordersoorten gedurende de werkweek getoond. Aangezien er geen interritten op zondag zijn, is er geen staaf op die dag in het histogram waarneembaar. Voor de overige dagen zijn maandag en zaterdag de rustigste dagen, terwijl dinsdag tot en met vrijdag de drukste dagen zijn.



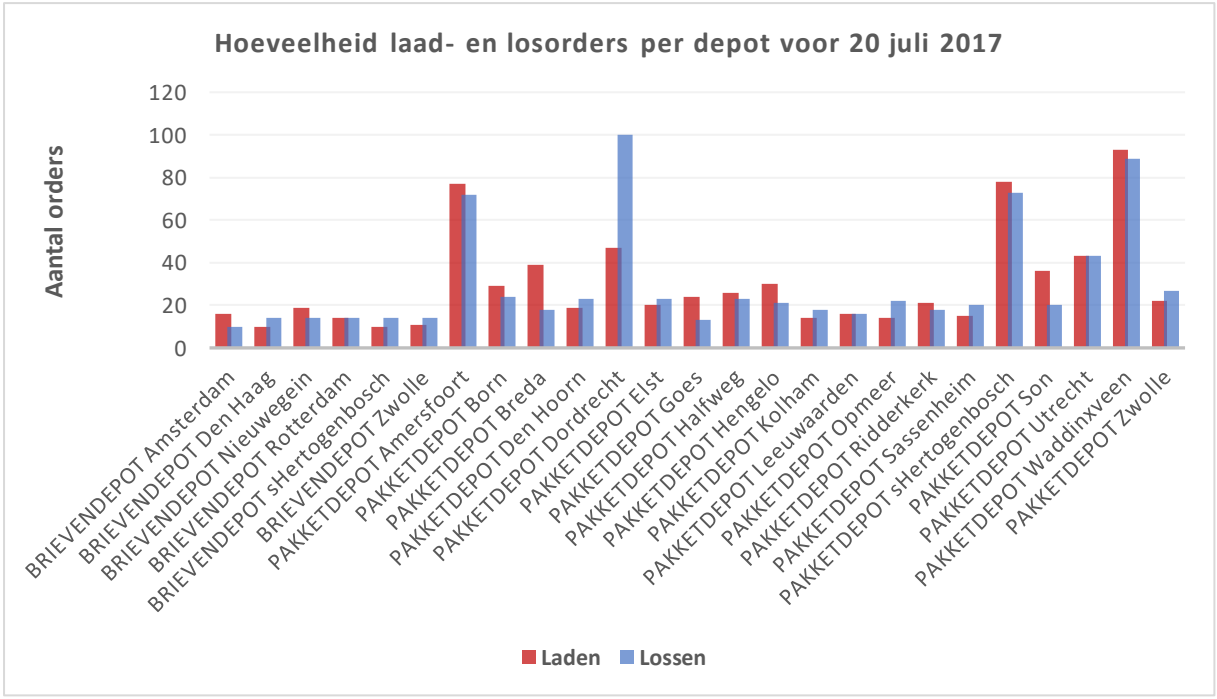
Figuur 4: Totaal aantal structurele orders met ordersoort 60 en 70 gedurende werkweek 29 2017

In totaal zijn er 25 PostNL depots verspreid in Nederland. Dit geeft een ongericht netwerk met 300 verbindingen ($\binom{25}{2} = 300$). Om inzicht te krijgen in de verhoudingen tussen het aantal laad- en lostransporten tussen de depots gedurende werkweek 29 is Figuur 5a gemaakt. Opvallend in dit figuur zijn de vier uitschieters van de pakketdepots (Amersfoort, Dordrecht, 's-Hertogenbosch en Waddinxveen). Op Dordrecht na vallen de overige drie pakketdepots onder de categorie depot+. In een depot+ worden orders met een te verre eindlocatie opnieuw gesorteerd en vervolgens geladen in een vrachtwagen richting eindlocatie. Voor pakketdepot Dordrecht is te zien dat de laad- en losopdrachten niet in balans zijn. Dit komt mede doordat er veel losopdrachten in de avond zijn voor Dordrecht (collectie van klanten) en overdag weer meer laadopdrachten op depots (ophalen van

emballage). Ook een mogelijke invloed op deze onbalans is dat alle pakketten voor België via Dordrecht gaan. In Figuur 5b is een soortgelijke grafiek gemaakt, maar dan voor de donderdag uit de werkweek. In deze figuur is hetzelfde patroon waarneembaar als in Figuur 5a. De andere dagen hebben een vergelijkend patroon.



Figuur 5a: Hoeveelheid laad- en losorders per depot in werkweek 29 2017



Figuur 5b: Hoeveelheid laad- en losorders per depot voor 20 juli 2017

3.3 Aannames

Om de complexiteit voor het inroosteren van de structurele orders met de tijdsvensters voor het laden en lossen tussen de depots in te perken en ervoor te zorgen dat er gelijke grenzen worden gedefinieerd voor het exacte, heuristische en het meta-heuristische model worden enkele aannames gedaan. Naar aanleiding van de databeschrijving en het data-onderzoek uit paragraaf 3.1 en 3.2 zullen eerst de aannames beschreven worden die een aanpassing op de ontvangen data van PostNL vereisen, waarna de resterende aannames later puntsgewijs zullen volgen.

Aangezien niet alle verbindingen tussen de depots in de dataset voorkwamen is de procestijd voor elke order opnieuw berekend. Door deze berekening kan het mogelijk gemaakt worden nieuwe routes tussen de orders te genereren. De rijtijden tussen depots zijn met behulp van Google Maps berekend voor een personenauto en vermenigvuldigd met een factor, zodat het kan voldoen aan de rijtijd van een vrachtwagen. Er is gekozen voor een factor 1,4 die overeenkomt met de meeste procestijden (minus de laad- en lostijden) van de al bekende orders uit de dataset. Naast de rijtijden tussen de depots en de attributen van de orderdata speelt de arbeidsduur van een chauffeur ook een belangrijke factor voor het vormen van een transportrooster. Volgens de gedefinieerde PostNL arbeidstijden moet een chauffeur in ieder geval acht uur lang werken tot een maximaal mogelijke arbeidstijd van twaalf uur per dag. Onder arbeidstijd vallen alle gewerkte uren met uitzondering van de pauze en voorziene wachttijd (bijvoorbeeld op de locatie waar de vrachtwagen wordt geladen of gelost). Met hierop volgend de resterende aannames:

1. Tijdsinterval van een dagplanning is tussen 20:00 – 8:00 uur.
2. Alle vrachtwagens zijn hetzelfde, in de zin dat er evenveel rolcontainers in passen.
3. Vrachtwagens starten en eindigen bij hun toegewezen startdepot.
4. Vrachtwagens starten leeg bij hun toegewezen startdepot.
5. Activiteiten per vrachtwagen:
    a. Een order uitvoeren. Dat wil zeggen: het gehele proces van het laden, rijden en lossen van een order,
    b. Leeg rijden,
    c. Leeg wachten.
6. Rijtijd en afstand tussen locaties is constant.
7. De vaste duur per laad- of losactiviteit is 25 minuten.
8. Procestijd = laadtijd + rijtijd + lostijd.
9. Chauffeurs hebben een arbeidstijd van minimaal 8 uur tot een maximum van 12 uur per dag.
10. Pauzes zijn in de wachttijd verwerkt.

4 ILP-model

Dit hoofdstuk zal beginnen met de beschrijving van de formulering van het gebruikte exacte ILP-model. Daarna zal beschreven worden hoe dit model geïmplementeerd is met daarbij een voorbeeld voor een testmodel met de benodigde input en gewenste output.

4.1 Formulering

Om te bepalen in hoeverre het mogelijk is om het gestelde probleem exact op te lossen, wordt er gebruik gemaakt van *Integer Linear Programming (ILP)*. ILP is een methode waarbij de optimale waarde van een lineaire functie over meerdere beslissingsvariabelen wordt bepaald om een lineair optimalisatieprobleem op te kunnen lossen, of de onoplosbaarheid ervan vast te kunnen stellen. Elk lineair optimalisatieprobleem is daarbij opgebouwd uit de volgende drie kernelementen: een doelfunctie, beslissingsvariabelen en randvoorwaarden. Voor het benodigd model voor PostNL zijn deze elementen verwerkt in onderstaande modelformulering, gebaseerd op die Van Barneveld [22] en Neumann [15]. In het vervolg van deze paragraaf zal systematisch de opbouw van het gemaakte ILP beschreven worden; startend met de nomenclatuur en notatie, gevolgd door de wiskundige probleemformulering, de doelfunctie en de restricties (beslissingsvariabelen met randvoorwaarden).

Nomenclatuur

Verzamelingen

D	Depots
O	Orders
V	Vrachtwagens

Parameters

A^{min}	Minimale arbeidstijd per dag
A^{max}	Maximale arbeidstijd per dag
a_0^+	Vroegst mogelijke laadtijd voor order *o*
a_0^-	Vroegst mogelijke lostijd voor order *o*
b_0^+	Laatst mogelijke laadtijd voor order *o*
b_0^-	Laatst mogelijke lostijd voor order *o*
M	Groot getal
p_o	Procestijd van order *o*
r_{ij}	Rijtijd tussen depot *i* en *j*
t	Tijd

Dummy's

o_v^{begin}	Beginorder van vrachtwagen *v*
o_v^{eind}	Eindorder van vrachtwagen *v*
o^{struct}	Structurele order

Nomenclatuur (vervolg)

Variabelen

X_{ovt}	$= \begin{cases} 1, & Als\ order\ o\ start\ op\ tijdstip\ t\ en\ wordt\ uitgevoerd\ door\ vrachtwagen\ v \\ 0, & Anders \end{cases}$	
U_v	$= \begin{cases} 1, & Als\ vrachtwagen\ v\ wordt\ ingezet\ voor\ minstens\ één\ order \\ 0, & Anders \end{cases}$	
$G_{o,o_v^{eind}}$	$= \begin{cases} 1, & Als\ order\ o\ en\ de\ eindorder\ van\ vrachtwagen\ v\ zich\ in\ dezelfde\ route\ bevinden \\ 0, & Anders \end{cases}$	
$H_{o,o_v^{begin}}$	$= \begin{cases} 1, & Als\ order\ o\ en\ de\ beginorder\ van\ vrachtwagen\ v\ zich\ in\ dezelfde\ route\ bevinden \\ 0, & Anders \end{cases}$	

Constanten

y_{ot}	$= \begin{cases} 1, & a_0^+ \leq t \leq b_0^+ \\ M, & Anders \end{cases}$	Laden
z_{ot}	$= \begin{cases} 1, & a_0^- \leq t \leq b_0^- \\ M, & Anders \end{cases}$	Lossen

4.1.1 Wiskundige probleemformulering

Het ongerichte logistieke netwerk voor de verzameling van *m* orders bestaat uit 25 PostNL brieven- en pakketdepots, die we achtereenvolgens definiëren als $O = \{o_1, o_2, \ldots, o_m\}$ en $D = \{d_1, d_2, \ldots, d_{25}\}$. Hierbij bestaat elke order *o* uit de volgende elementen: een ordernummer, een procestijd ($p_o \in \mathbb{N} \cup \{0\}$), een laad- en loslocatie, en een tijdsvenster voor het laden ($[a_0^+, b_0^+]$) en lossen ($[a_0^-, b_0^-]$). Een order is succesvol uitgevoerd als de vracht vertrekt van het laaddepot en arriveert bij zijn losdepot binnen zijn tijdsvenster. Dat wil zeggen: na zijn vroegst mogelijke lostijd of voor zijn laatst mogelijke lostijd om de vracht volledig gelost te kunnen hebben.

Voor het succesvol voltooien van alle structurele orders ($\forall o^{struct}$) tussen de depots is een verzameling van *n* vrachtwagens beschikbaar, waarvoor we definiëren $V = \{v_1, \ldots, v_n\}$. Elke vrachtwagen die ingezet wordt, begint en eindigt bij zijn vooraf toegewezen depot. Voor het realiseren van deze toewijzing wordt er per vrachtwagen een beginorder o_v^{begin} en een eindorder o_v^{eind} aangemaakt met een procestijd van nul, waarbij geldt dat *begin* gelijk is aan *eind*. Een ingezette vrachtwagen kan de volgende activiteiten uitvoeren: laden, lossen, wachten of tussen twee depots rijden (met één structurele order of leeg). Elke ingezette vrachtwagen moet voldoen aan de arbeidstijd van een chauffeur per dag met het interval $[A^{min}, A^{max})$, waarbij het domein $\mathbb{N} \cup \{0\}$ betreft. De procestijd van een order *o* is de totale tijd die nodig is om de structurele order *o* te laden, rijden en lossen. De rijtijd ($r_{ij} \in \mathbb{N} \cup \{0\}$) is gedefinieerd als de tijd die een vrachtwagen nodig heeft om van depot *i* naar depot *j* te komen. Voor de tijdshorizon definiëren we tijd *t* met het domein $\mathbb{N} \cup \{0\}$. Deze parameter

loopt van de begintijd tot en met de eindtijd van het transportproces. De constante y_{ot} geeft voor elke order *o* aan op welk tijdstip *t* er wel (waarde 1) of niet (waarde M) geladen kan worden, en analoog voor de constante z_{ot} voor het lossen. De variabele X_{ovt} krijgt waarde 1 als order *o* met vrachtwagen *v* gekoppeld is op tijdstip *t*. Anders wordt er een waarde van nul toegekend aan X_{ovt}. Terwijl variabele U_v de waarde 1 krijgt als vrachtwagen *v* wordt ingezet voor minstens één structurele order. Anders wordt hier ook een waarde van nul aan U_v toegekend. Ook zijn de variabelen $G_{o,o_v^{eind}}$ en $H_{o,o_v^{begin}}$ gedefinieerd om ervoor te zorgen dat de structurele orders altijd tussen de begin- en eindorder worden geplaatst. Indien order *o* en de beginorder/eindorder van vrachtwagen *v* zich in dezelfde route bevinden krijgt de betreffende variabele waarde M. Anders krijgt deze variabele waarde nul. Het uiteindelijk doel van het ILP-model is om de totale ingezette tijd voor het inroosteren van alle structurele orders te kunnen doen minimaliseren.

4.1.2 Doelfunctie

Het minimaliseren van de totale ingeroosterde tijd van alle ingezette vrachtwagens bij elkaar:

$$\min \sum_{v \in V} \left(\sum_{o_v^{eind} \in O_v^{eind}} \sum_{t_1=0}^{T_{max}} t_1 \, X_{o_v^{eind} v t_1} - \sum_{o_v^{begin} \in O_v^{begin}} \sum_{t_2=0}^{T_{max}} t_2 X_{o_v^{begin} v t_2} \right) \qquad (1)$$

4.1.3 Restricties

- Iedere structurele order *o* moet eenmalig uitgevoerd worden door een vrachtwagen gedurende het tijdsinterval *t,* rekening houdend met de vooraf bekende tijdsvensters van het laden en lossen:

$$\sum_{t=0}^{T_{max}} \sum_{v \in V} X_{ovt} * y_{ot} * z_{o(t+p_o)} = 1, \qquad \forall o \in o^{struct} \qquad (2)$$

- Order *o* moet eerst afgerond zijn door vrachtwagen *v* voordat deze met een vervolgorder (*o'*) kan starten, rekening houdend met de rijtijd tussen het losdepot van order *o* en het laaddepot van order *o'*:

$$X_{ovt} + \sum_{s=0}^{p_o + r_{ij} - 1} X_{o' v (t+s)} \leq 1,$$
$$\forall (o, o') \in \left\{ o_v^{begin} \right\}_{v \in V} \cup \left\{ o_v^{eind} \right\}_{v \in V} \cup o^{struct} \qquad (3)$$
$$met \; o \neq o', \forall v \in V, \qquad t = [0, \dots, T_{max}]$$

- Restrictie die voorkomt dat er geen structurele orders voor de beginorder van vrachtwagen *v* geplaatst kunnen worden:

$$G_{o,o_v^{eind}} \leq M * \left(1 - \sum_{t=0}^{Tmax} X_{ovt} \right), \qquad \forall o \in \left\{ o_v^{begin} \right\}_{v \in V} \cup o^{struct}, \forall v \in V \qquad (4)$$

- Restrictie die voorkomt dat er geen structurele orders na de eindorder van vrachtwagen *v* geplaatst kunnen worden:

$$H_{o,o_v^{begin}} \leq M * \left(1 - \sum_{t=0}^{Tmax} X_{ovt}\right), \qquad \forall o \in \left\{o_v^{eind}\right\}_{v \in V} \cup o^{struct}, \forall v \in V \tag{5}$$

- Beginorder o_v^{begin} moet als eerst en eindorder o_v^{eind} als laatst uitgevoerd worden voor iedere vrachtwagen *v*:

$$\sum_{t_1=0}^{T_{max}} t_1 \, X_{o_v^{begin} v t_1} \leq H_{o,o_v^{begin}} + \sum_{t_2=0}^{T_{max}} t_2 \, X_{ovt_2}, \quad \left\{o_v^{begin}\right\}_{v \in V}, \forall o \in \left\{o_v^{eind}\right\}_{v \in V} \cup o^{struct}, \forall v \in V \tag{6}$$

en

$$\sum_{t_1=0}^{T_{max}} t_1 \, X_{o_v^{eind} v t_1} \geq \sum_{t_2=0}^{T_{max}} (t_2 + p_o) \, X_{ovt_2} - G_{o,o_v^{eind}}, \quad \{o_v^{eind}\}_{v \in V}, \forall o \in \left\{o_v^{begin}\right\}_{v \in V} \cup o^{struct}, \forall v \in V. \tag{7}$$

- Iedere vrachtwagen moet zijn begin- en eindorder uitvoeren:

$$\sum_{t=0}^{T_{max}} X_{o_v^{begin} v t} = 1, \qquad \left\{o_v^{begin}\right\}_{v \in V}, \forall v \in V \tag{8}$$

en

$$\sum_{t=0}^{T_{max}} X_{o_v^{eind} v t} = 1, \qquad \{o_v^{eind}\}_{v \in V}, \forall v \in V. \tag{9}$$

- Afdwingen dat $U_v = 1$ als vrachtwagen *v* ingezet wordt voor minstens één structurele order:

$$\sum_{t=0}^{T_{max}} \sum_{o \in o^{struct}} X_{ovt} \leq M * U_v, \quad \forall v \in V \tag{10}$$

- Iedere chauffeur toegewezen aan een vrachtwagen moet tenminste voor een arbeidstijd van A^{min} ingezet worden:

$$\sum_{t_1=0}^{T_{max}} X_{o_v^{eind} v t_1} - \sum_{t_2=0}^{T_{max}} t_2 X_{o_v^{begin} v t_2} \geq A^{min}, \quad \left\{o_v^{begin}\right\}_{v \in V}, \left\{o_v^{eind}\right\}_{v \in V}, \forall v \in V \tag{11}$$

- Iedere chauffeur toegewezen aan een vrachtwagen mag niet langer da een arbeidstijd van A^{max} ingezet worden:

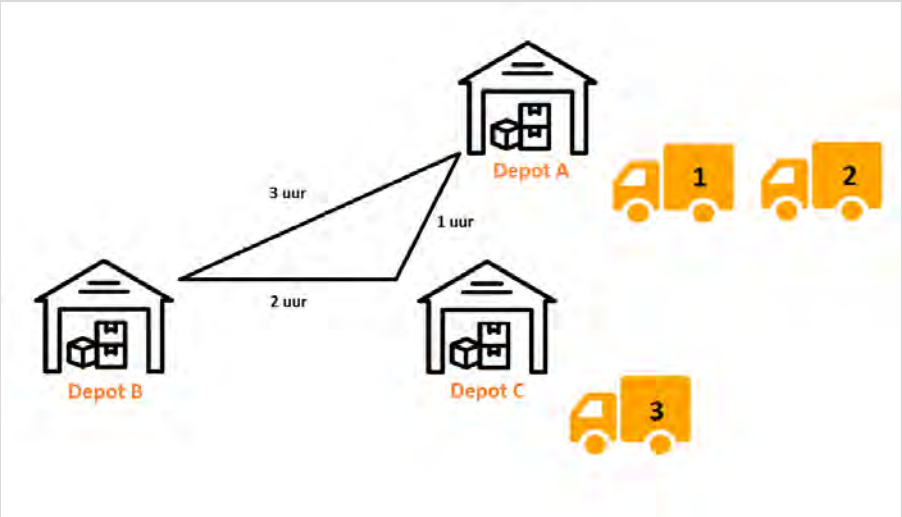
$$\sum_{t_1=0}^{T_{max}} t_1 X_{o_v^{eind} v t_1} - \sum_{t_2=0}^{T_{max}} t_2 X_{o_v^{begin} v t_2} \leq A^{max}, \quad \left\{o_v^{begin}\right\}_{v \in V}, \left\{o_v^{eind}\right\}_{v \in V}, \forall v \in V \tag{12}$$

- Randvoorwaarde van de binaire variabele X_{ovt} definiëren:

$$X_{ovt} \in \{0,1\}, \quad \forall o \in \left\{o_v^{begin}\right\}_{v \in V} \cup \left\{o_v^{eind}\right\}_{v \in V} \cup o^{struct}, \quad \forall v \in V, t = [0, \dots, T_{max}] \tag{13}$$

4.2 Implementatie

Voor de implementatie van het ILP-model is er gekozen voor het gebruik van het Python *package* PuLP. Met deze lineaire *solver* is het mogelijk om ILP bestanden te genereren die kunnen worden ingelezen door andere *solvers*. Aangezien PuLP tijdens de implementatie niet de snelste *solver* bleek te zijn, is er gekozen om de gemaakte restricties door te geven aan een snellere solver CPLEX. Om te testen of het ILP-model werkt is er eerst een klein databestand gemaakt met zes orders en drie depots (zie hiervoor Figuur 6 en Tabel 3). Hierbij starten twee vrachtwagens bij depot A en één vrachtwagen bij depot C.



Figuur 6: Logistieke netwerk voor het ILP-testmodel

	Laden		Lossen	
Order	Locatie	Tijdsvenster (in uren)	Locatie	Tijdsvenster (in uren)
1	Depot A	[0, 5)	Depot B	[0, 12)
2	Depot A	[1, 6)	Depot C	[1, 12)
3	Depot B	[0, 5)	Depot A	[0, 12)
4	Depot B	[3, 8)	Depot A	[3, 12)
5	Depot C	[1, 6)	Depot A	[1, 12)
6	Depot C	[3, 8)	Depot B	[3, 12)

Tabel 3: Orderdata voor het ILP-testmodel

Om het controleren te vergemakkelijken is in dit voorbeeld ervoor gekozen om de laad- en lostijden buiten beschouwing te houden, waardoor de procestijd gelijk is aan de rijtijd tussen de betreffende depots. Het datavoorbeeld resulteert in de optimale oplossing van het gestelde probleem in het rooster in Tabel 4 en de Gantt-grafiek in Figuur 7. In de oplossing is de totaal ingeroosterde tijd voor de twee toegewezen vrachtwagens 16 uur, waarbij vrachtwagen 2 van depot A niet wordt ingezet. Dit resultaat is optimaal, omdat de orders zeer efficiënt zijn ingedeeld (mooi op elkaar aansluiten). Het enige onontkoombare is de gemaakte wachttijd van 2 uur om aan de minimale arbeidstijd voor de chauffeur van vrachtwagen 1 te voldoen.

	Rooster		
	Order	Starttijd	Eindtijd
Vrachtwagen 1	Begin	0	0
	1	0	3
	4	3	6
	Eind	8	8
Vrachtwagen 3	Begin	1	1
	5	1	2
	2	2	3
	6	3	5
	3	5	8
	Eind	9	9

Tabel 4: Output van de oplossing van het ILP-testmodel

Figuur 7: Gantt-grafiek voor het ILP-testmodel met de begin-en eindorders (zwart), structurele orders (oranje) met ordernummers erin vermeld en de wachttijd (grijs)

5 Heuristiek & Meta-heuristiek

In dit hoofdstuk worden de methoden beschreven die zijn gebruikt om een geldig rooster[2] te kunnen vinden voor alle structurele orders van PostNL met ordersoort 60 en 70 van 20 juli tot 21 juli 2017. Hierbij wordt voor de heuristische methode het *Concurrent Scheduler Approach (CSA)* beschreven en voor de meta- heuristische methode het *Genetic Algorithm (GA)*.

5.1 Concurrent Scheduler Approach

De *Concurrent Scheduler Approach (CSA)* is een *greedy* heuristiek die eenvoudig en snel een toelaatbare oplossing kan vinden voor het *Vehicle Routing Problem with Time-Windows*. Deze methode beschreven in [10] doorloopt de volgende stappen:

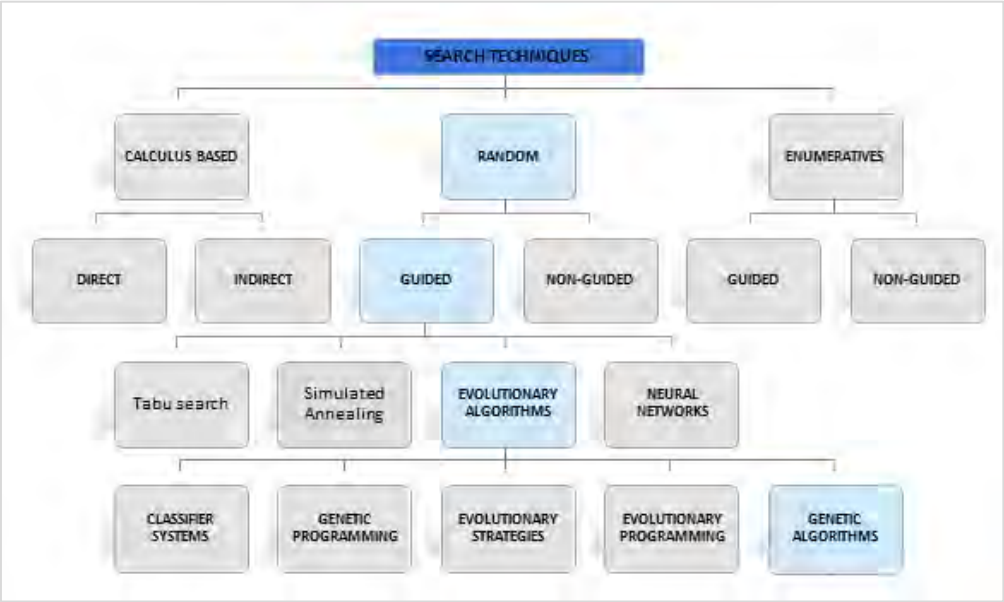
1. Sorteer alle orders op starttijd.
2. Wijs de eerste order toe aan vrachtwagen 1.
3. Ga voor de overige orders na of het mogelijk is om de eerstvolgende order toe te wijzen aan een al ingezette vrachtwagen. Indien mogelijk voer stap (a) uit, anders stap (b):
    a. Voeg de order toe aan de vrachtwagen met de laagste *deadhead time.*
    b. Voeg de order toe aan een nieuwe vrachtwagen.

Met de *deadhead time* wordt de onbenutte tijd van een vrachtwagen bedoeld gedurende een moment in het proces. Per vrachtwagen wordt er dan gekeken wanneer een vrachtwagenchauffeur klaar is met het uitvoeren van op dat moment zijn laatst ingeplande order, om vervolgens de onbenutte tijd van de betreffende vrachtwagen te berekenen. Hierbij kan de onbenutte tijd berekend worden door na te gaan hoelang het duurt om de vrachtwagen van zijn huidige locatie naar de laadlocatie van de vervolgorder te rijden, of hoelang het duurt om leeg naar zijn einddepot te rijden. De minimale waarde van de reistijd naar de volgende locatie of einddepot bepaalt de laagste *deadhead time* voor de betreffende vrachtwagen. Bij de toewijzing van een order aan een nieuwe vrachtwagen wordt als vertrekpunt de laadlocatie gekozen van de betreffende order. Om het beschreven CSA toe te passen op het probleem voor deze scriptie zal er naast de tijdsvensters ook rekening gehouden moeten worden met eventuele wachttijden en arbeidstijden van een chauffeur. Indien de berekende arbeidstijd van een vrachtwagenchauffeur onder de minimale arbeidsduur van acht uur valt, zal de resterende tijd als wachttijd erbij opgeteld worden. Verder zal er voor het toewijzen van de eerstvolgende order gekeken worden of de vrachtwagenchauffeur binnen de maximale arbeidstijd van twaalf uur valt door te controleren of het mogelijk is om na het uitvoeren van de vervolgorder terug te rijden naar het einddepot. Indien dat niet mogelijk is zal de order aan een nieuwe vrachtwagen gegeven worden (stap 3b.). Net zoals voor het ILP-model is de waarde voor de doelfunctie te berekenen door de totale ingeroosterde tijd bij elkaar op te tellen van alle ingezette vrachtwagens in het verkregen rooster.

[2] Een rooster dat voldoet aan de tijdsventers voor het laden en lossen en rekening houdt met de arbeidstijden en voorwaarden van een vrachtwagenchauffeur

5.2 Genetic Algorithm

Een *Genetic Algorithm (GA)* is een *random*-gestuurde zoektechniek die behoort tot een van de varianten van de evolutionaire algoritmen (zie Figuur 8). Dit algoritme vindt zijn oorsprong in de kunstmatige intelligentie en wordt vaak gebruikt voor het vinden van oplossingen voor zoek- en optimalisatieproblemen.



Figuur 8: Taxonomie van zoektechnieken, van [21] aangepast voor de doelstelling van deze scriptie

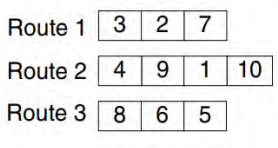
Het onderliggende idee om evolutie toe te passen is gebaseerd op het fenomeen natuurlijke selectie: de drang dat alleen de beste individuen van een populatie in een omgeving van schaarste kunnen overleven om zich te kunnen reproduceren. Hierbij is in termen van het algoritme een individu een mogelijke oplossing van het probleem. Het verschil met GA ten opzichte van de andere evolutionaire algoritmen is dat een individu bestaat uit een string van gehele getallen, terwijl bijvoorbeeld *Genetic Programming* daarvoor een boomstructuur heeft en *Evolutionary Strategies* reële waarden daarvoor hanteert. Voor het volledig kunnen laten *runnen* van het GA is het belangrijk om te weten waaruit een evolutionair algoritme is opgebouwd. In Figuur 9 is het algemeen schema beschreven van de stappen die het GA zou moeten maken. Voor het vervolg van dit hoofdstuk worden deze stappen gedetailleerd beschreven om uiteindelijk tot de vorming van het gebruikte algoritme te komen.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Figuur 9: Algemeen schema van een pseudo code voor een evolutionair algoritme, van [6]

5.2.1 Initialization

Om de beginpopulatie van individuen te kunnen initialiseren zal eerst de representatie van een individu geformuleerd moeten worden. Zoals eerder beschreven is een individu een oplossing van het probleem, voor deze scriptie is dat een rooster die per ingezette vrachtwagen beschrijft in welke volgorde de orders uitgevoerd kunnen worden. Bij het initialiseren van de individuen wordt het aantal vrachtwagens en orders per vrachtwagen, niet vastgezet. Deze variabelen kunnen namelijk in de vervolgstappen van het GA nog veranderen. Voor de representatie van een individu is er gebruik gemaakt van de *Genetic Vehicle Representation (GVR)* in Periera et al. [16]. In Figuur 10 is een voorbeeld hiervan zichtbaar, waarin een twee-level schema te zien is die alle informatie omvat voor een oplossing van tien orders met drie ingezette vrachtwagens. In dit schema met twee lagen wordt in de verticale richting de routenummers van de ingezette vrachtwagens uitgezet en in de horizontale richting wordt voor elke route de ordervolgorde weergegeven. In deze representatie bepaalt de startorder de locatie van het begin- en einddepot. Volgens [16] is deze methode robuust en effectief voor het vinden van betere nieuwe oplossingen voor de al bekende benchmark problemen voor het *Vehicle Routing Problem (zie hiervoor hoofdstuk 2)*.



Figuur 10: Representatie van een individu (rooster) voor tien orders met drie ingezette vrachtwagens, van [*16] aangepast voor de doelstelling van deze scriptie*

De initialisatie van de beginpopulatie wordt gedaan door gebruik te maken van de verkregen oplossing van de *Concurrent Scheduler Approach (CSA)*. Verder zal de populatie worden aangevuld met het *random* aan laten maken van geldige roosters, zodat het GA begint met alleen toelaatbare oplossingen. De gebruikte methode lijkt op die van het CSA, maar verschilt in het feit dat de orders *random* worden toegewezen. Daarnaast heeft telkens maar één vrachtwagen de mogelijkheid om deze order toegewezen te krijgen, terwijl bij het CSA meerdere vrachtwagens op dat moment daarvoor beschikbaar kunnen zijn. De methode begint net als het CSA met het sorteren van de orderdata op starttijd en het toewijzen van de eerste order aan vrachtwagen 1. Daarna wordt er *random* een nog niet toegewezen order uitgekozen. Indien deze gekozen order resulteert in een toelaatbare oplossing, dan zal deze order toegevoegd worden aan de beschikbare vrachtwagen. Vervolgens wordt nogmaals gezocht naar een nieuwe order die kan worden toegevoegd. Voor het geval er geen toelaatbare oplossing wordt gevonden, zal er een aantal *random* pogingen gedaan worden om andere orders toe te kunnen voegen. Indien dat ook geen toelaatbare oplossing genereert, wordt de huidige vrachtwagen als volledig gevuld gezien en wordt er een nieuwe vrachtwagen aan de eerstvolgende order toegewezen. Dit gehele proces herhaalt zich totdat alle orders zijn toegewezen om een compleet rooster te vormen. Voor de bepaling van het aantal roosters (de populatiegrootte) dat aangemaakt moet worden is er gekozen voor een constante waarde van 100 om voldoende diversiteit in de populatie te behouden. Deze waarde wordt vrijwel altijd constant gehouden in evolutionaire algoritmen, zodat de competitie voor de beperkte bronnen in een omgeving staande gehouden kan worden [6].

5.2.2 Evaluation

In de evaluatiefase krijgt elk individu $\bar{x}$ van de populatie een waarde toegekend [6]. Deze waarde wordt berekend met de *evaluation function eval($\bar{x}$)* die de basis vormt voor de twee selectie rondes in het GA (*parentselection* en *survivor selection)*. Hierbij hangt de kwaliteit van de waarde uit de *evaluation function* af of het om een minimalisatie of maximalisatie probleem gaat. Indien het een minimalisatie probleem betreft, dan wordt een zo laag mogelijke waarde hiervan als goed gezien en omgekeerd. Om het beoogd probleem van PostNL uit te drukken in een *evaluation function* is het belangrijk om na te gaan wat de definitie van een goed rooster is. Hierbij wordt de waarde van een rooster net zoals het ILP-model en het CSA uitgedrukt in de totale ingeroosterde tijd van alle ingezette vrachtwagens. Indien een route van een vrachtwagen onder de minimale arbeidstijd valt wordt er de waarde 8 aan toegekend. Dit resulteert voor vrachtwagen *v* in de *objective function* $f_v(\bar{x})$ geformuleerd in functie 14, waarbij A_v de arbeidstijd van de betreffende chauffeur is.

$$f_v(\bar{x}) = \max(A_v, 8) \tag{14}$$

Tijdens het evolutieproces is er een kans aanwezig dat er ongeldige roosters ontstaan, waarbij tijdsvensters en/of de maximale arbeidstijd van een route worden overschreden. Aangezien er een hoge prioriteit in het transportproces van PostNL is om aan deze restricties te voldoen, is ervoor gekozen om een *penalty function* $p(\bar{x})$ te gebruiken. Hierbij is de *penalty* voor het overtreden van de tijdsvensters in een route van vrachtwagen *v* gedefinieerd als $p_v(\bar{x})^{\text{tijdsvensters}}$ en de *penalty* voor het overschrijden van de maximale arbeidstijd van de chauffeur van vrachtwagen *v* als $p_v(\bar{x})^{\max\_arbeidstijd}$. Aangezien het beoogd probleem van PostNL een minimalisatie probleem betreft is de waarde van de *penalty functions* nul als aan de restricties wordt voldaan en anders positief. Zie functie 15 en 16 voor de formulering van de gebruikte *penalty functions:*

$$p_v(\bar{x})^{\text{tijdsvensters}} = \begin{cases} 1, & De\ tijdsvensters\ worden\ door\ vrachtwagen\ v \\ & overschreden \\ 0, & Anders \end{cases} \tag{15}$$

en

$$p_v(\bar{x})^{\text{max\_arbeidstijd}} = \begin{cases} 1, & De\ maximale\ arbeidstijd\ van\ de\ chauffeur \\ & van\ vrachtwagen\ v\ wordt\ overschreden \\ 0, & Anders \end{cases} \tag{16}$$

Dit resulteert in de geformuleerde *evaluation function* 17, waarbij W_1 en W_2 constanten zijn die de hoogte van de penalty bepalen. Hierbij is er gekozen om deze constanten hoog (met elk waarde 1000) te nemen voor het tegengaan van het gebruik van ontoelaatbare oplossingen.

$$eval(\bar{x}) = \sum_{v \in V} (f_v(\bar{x}) + W_1 * p_v(\bar{x})^{\text{tijdsvensters}} + W_2 * p_v(\bar{x})^{\max\_arbeidstijd}) \tag{17}$$

5.2.3 Termination condition

In [6] worden verschillende methoden beschreven voor het beëindigen van het evolutieproces om te voorkomen dat het algoritme nooit zal kunnen stoppen met *runnen*. Indien de optimale waarde van de doelfunctie vooraf bekend zou zijn, dan kan deze waarde als stopcriterium dienen. Aangezien het optimum van het beschreven probleem onbekend is, zal er een ander stopcriterium moeten gaan gelden. Vanwege de stochastische aard van een evolutionair algoritme is er geen garantie op het vinden van een betere oplossing, maar met een groot aantal iteratiestappen (de geselecteerde roosters voor de nieuwe generatie) zou deze kans vergroot kunnen worden. Daarom is er voor het gebruikte GA gekozen om een limiet van 1000 iteratiestappen in te stellen als stopcriterium.

5.2.4 Parentselection

De *parentselection* is het gedeelte waar de individuen uit de huidige populatie worden gekozen op basis van hun waarde uit de evaluatiefase [6]. Deze individuen worden ook wel de *parents* genoemd, die zorgen ervoor dat er *offspring* gecreëerd kan worden in de *recombination* fase en *mutation* fase. Op basis van [11] is er gekozen om voor het GA een *tournament selection* te doen met een klein aantal deelnemers van twee, om diversiteit in de populatie te behouden. In een *tournament selection* worden eerst *random* individuen uit de huidige populatie gekozen. Vanuit de selectie van individuen wordt de beste van deze individuen als *parent* gekozen. Dit proces herhaalt zich totdat het gewenste aantal *parents* van 100 bereikt is. In de *tournament selection* kunnen dezelfde individuen weer opnieuw verkozen worden.

5.2.5 Variation operators (recombination & mutation)

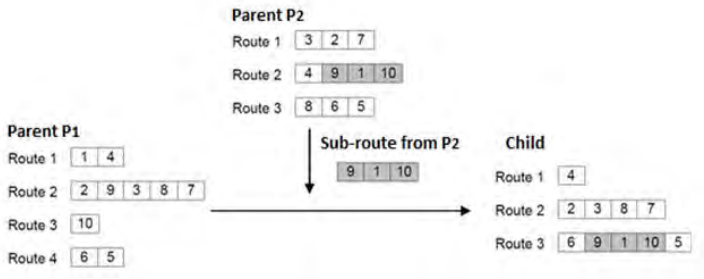
Samen met *mutation* maakt *recombination* deel uit van de variatie operatoren. Deze operatoren zorgen ervoor dat er nieuwe individuen gevormd kunnen worden uit de huidige generatie [6]. Waar de operator *recombination* zorgt voor het creëren van *offspring* door de geselecteerde *parents* uit de *parentselection* te combineren, zorgt de operator *mutation* voor een kleine aanpassing op een geselecteerde *parent* of op de verkregen *offspring* uit de *recombination* fase. In Figuur 11 is het gebruikte algoritme uit [16] voor de *recombination* beschreven. Hiermee kan ervoor gezorgd worden dat een gedeelte van een rooster (*parent* P_1) doorgegeven kan worden aan een ander rooster (*parent* P_2), door gebruik te maken van een *sub-route (SR).* De gekozen reeks *S* omvat alle geselecteerde roosters uit de fase van de *parentselection*.

Voor elke *parent* P_1 van de geselecteerde reeks *S* **Herhaal**:

1. Selecteer *random* een andere *parent* P_2 van *S*.
2. Van het rooster van P_2 selecteer *random* een *sub-route SR*: $\{a_1, a_2, \ldots, a_n\}$.
3. Vindt de order *o,* die niet behoort tot *SR*, met de laagste rijdtijd naar a_1.
4. Voeg *SR* toe aan het rooster van P_1, zodanig dat a_1 direct na order *o* wordt geplaatst.
5. Verwijder uit het rooster van P_1 alle dubbele orders die ook voorkomen in *SR*. Dit resulteert in het rooster voor een *child*.

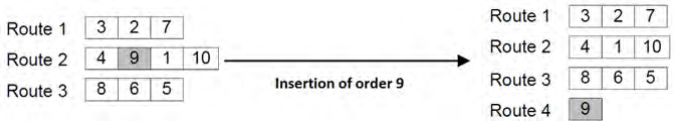
Figuur 11: Algoritme van de recombination met sub-route, van [16] aangepast voor de doelstelling van deze scriptie

In Figuur 12 is een voorbeeld van deze methode zichtbaar, waarbij voor de *offspring* (*child*) het aantal vrachtwagens zelfs met één gereduceerd is.



Figuur 12: Voorbeeld van recombination, van [16] aangepast voor de doelstelling van deze scriptie

Voor het gebruikte GA is er voor de *mutation* fase gekozen voor de methode *insertion* die wordt aanbevolen voor het *Vehicle Routing Problem with Time-Windows* door [11]. Hierbij wordt een order *random* uit het rooster gekozen om op een andere *random* geselecteerde plaats gezet te worden (zie Figuur 13). Zowel *recombination* als *mutation* hebben elk een kans om wel of niet uitgevoerd te worden. Deze kans wordt bepaald door steeds een willekeurig getal (uit een uniforme verdeling met bereik [0, 1]) te vergelijken met een vaste drempelwaarde voor de *recombination (recombination rate)* en *mutation* (*mutation rate*). Indien het *random* getal onder de drempelwaarde van de betreffende variatie operator valt, wordt deze operator uitgevoerd. Om te voorkomen dat een rooster niet volledig steeds wordt omgegooid is er gekozen voor een *recombination rate* van 0,5. De gebruikte *mutation rate* van 1/ (2 x *Q*) is gebaseerd op [16], waarbij *Q* het aantal vrachtwagens van de huidige oplossing is. Door gebruik te maken van deze *mutation rate* is het creëren van een nieuwe route in proportie met het aantal vrachtwagens dat ingezet is.



Figuur 13: Voorbeeld van insertion, van [16] aangepast voor de doelstelling van deze scriptie

5.2.6 Survivor selection

In de *survivor selection* worden de individuen gekozen om door te gaan naar de volgende generatie [6]. Om ervoor te zorgen dat de beste oplossingen behouden blijven is er gekozen voor de *Deterministic Elitist Replacement by* ($\mu + \lambda$)*.* Hierbij worden zowel de *parents* (μ) als de *offspring* (λ) meegenomen om geselecteerd te kunnen worden op basis van hun waarde uit de evaluatiefase. Gedurende de selectieronde wordt eerst een vast percentage van de beste individuen geselecteerd, daarna worden de resterende individuen *random* gekozen tot de nieuwe populatie gevuld is voor de volgende evolutie. Om het vroegtijdig convergeren naar een lokaal optimum tegen te gaan is het nodig om voldoende diversiteit in de populatie aanwezig te houden [6]. Dit kan door het percentage te verlagen, waardoor ook individuen met een slechtere waarde uit de evaluatiefase door kunnen gaan. Voor het gemaakte GA is ervoor gekozen om dit percentage op 75% in te stellen.

5.2.7 Diagnose tot de vorming van het GA

De kracht van de natuurlijke evolutie van een GA zit hem in een bepaalde stijl van het oplossen van een probleem – die van "*trial-and-error*" [6]. Dat wil zeggen: voor het vormen van een GA behoort het proces van het testen en bijstellen van de stappen in het GA en de daarvoor gebruikte parameters. De twee componenten die de basis vormen van een evolutionair systeem zijn: de variatie en de selectie operatoren [6]. De variatie operatoren *recombination* en *mutation* zorgen voor diversiteit in de populatie, om daardoor de nieuwheid te bevorderen. De selectie operatoren (*parentselection* en *survivor selection*) zorgen voor het verbeteren van de gemiddelde kwaliteit van de individuen in de populatie.

Door het zorgvuldig kiezen van de gebruikte parameters zou het GA gestuurd kunnen worden in het zoekproces naar betere oplossingen. Hierbij blijft er altijd een kans aanwezig dat tijdens het evolutieproces de gehele populatie (alle individuen) naar een lokaal optimum nadert, in plaats van naar het globaal optimum. Om verschillende fasen in het zoekproces te categoriseren wordt er vaak gebruik gemaakt van de termen *exploration* en *exploitation* [6]. Grofweg gezien is *exploration* het genereren van nieuwe individuen in nog niet geteste gebieden van de oplosruimte, terwijl *exploitation* nieuwe individuen genereert door te zoeken in de buurt van de geteste gebieden met goede oplossingen. Te veel van de eerste (bij een te hoge *recombination rate*) kan leiden tot inefficiënt zoeken, en te veel van het laatste (bij een te hoge *mutation rate*) kan er voor zorgen dat het zoekproces te snel de neiging heeft om één richting uit te gaan [5]. Indien de diversiteit in de populatie te snel verloren raakt en vast komt te zitten in een lokaal optimum spreekt men van *premature convergence* [6].

Wanneer een probleem restricties met zich mee neemt betekent dit dat niet alle mogelijke oplossingen toelaatbare oplossingen voor het probleem kunnen zijn. Hierdoor ontstaat er een verdeling van de potentiele oplossingen in een *feasible region (*of *regions)* en een *infeasible region* [6]. De eerste bevat de individuen die wel aan de restricties voldoen (toelaatbare oplossingen) en de laatste niet (ontoelaatbare oplossingen). Het omgaan met restricties in evolutionaire algoritmen wordt niet als makkelijk ondervonden, omdat de variatie operatoren typisch "blind" zijn voor deze restricties [6]. Dat wil zeggen: zelfs als de *parents* beide een toelaatbare oplossing zijn, is er geen garantie dat hun *offspring* daar ook aan gaat voldoen. Voor optimalisatie problemen met restricties (*Constrained Optimization Problems, COP)*, zoals het probleem van PostNL, wordt vaak een *penalty* aan de *evaluation function* toegevoegd. Een vaak gebruikte techniek voor dit soort problemen is de toevoeging van *static penalties* [13]. Drie methoden die regelmatig onder deze techniek toegepast worden zijn *extinctive penalties*, *binary penalties*, en *distance-based penalties*. De eerste betreft alle *penalty* constanten (W_i) hoog te zetten om het gebruik van ontoelaatbare oplossingen zoveel mogelijk tegen te gaan. De methode *binary penalties* geeft de waarde 1 terug als niet aan de restrictie is voldaan en omgekeerd de waarde nul. De laatste methode hanteert een afstandsmatrix die per restrictie de moeilijkheidsgraad van het probleem omvat. Uit [9] volgt dat deze laatste methode de beste resultaten geeft. Toch blijft de bepaling van ieder van de *penalty* waarden het grootste probleem. Indien er specifieke domeinkennis van het probleem is zou in sommige situaties na meerdere *runs* deze waarden bepaald kunnen worden, maar dit is een tijdrovend proces [6]. Aangezien het genereren van het GA bij het eindstadium van dit afstudeerproject naar voren kwam, was er onvoldoende tijd om per restrictie verschillende *penalty* waarden uitgebreid te kunnen testen. Daarom is voor de eerste methode gekozen die alle *penalty* constanten een hoge waarde toekend.

Gedurende het *runnen* blijkt dat het genereren van oplossingen waar alle orders aan routes worden gekoppeld niet het tijdrovende gedeelte van het GA is – het optimaliseren van orders naar een geldig rooster met een zo min mogelijke ingeroosterde tijd wel. Na vele testen met verschillende parameterinstellingen wordt er zelden in de *recombination* fase en de *mutation* fase een toelaatbare oplossing als *offspring* gecreëerd. Dit komt doordat de *random* gekozen *sub-route* of order (zie paragraaf 5.2.5) meestal op een plek geplaatst wordt bij orders die er geen order tussen kunnen hebben, wat leidt tot een ongeldige route en daardoor ook een ongeldig rooster. In dit gedeelte van het proces blijkt bij gebruik van alleen de variatie operator *mutation* er vaker toelaatbare oplossingen gecreëerd worden. Wel zijn de beperkt aantal gecreëerde oplossingen bij zowel *mutation* met of zonder *recombination* niet beter dan de oplossingen van de beginpopulatie (één rooster van het CSA en de rest *random* aangemaakt). Hierdoor blijft na een aantal opeenvolgende generaties de diversiteit in de populatie sterk afnemen, waarbij de gehele populatie na een korte tijd *runnen* naar een lokaal optimum leidt (*premature convergence*).Het is zelfs zo dat de gehele populatie van het GA gelijk is aan één van de al gegenereerde oplossingen uit de beginpopulatie.

5.2.8 Samenvatting van het voorgestelde GA

In Tabel 5 worden de geselecteerde parameters weergegeven die gebruikt worden voor het *runnen* van het GA.

	Algoritmebeschrijving
Representation	*Genetic Vehicle Representation (GVR)*
Recombination	*Recombination* met *sub-route* met *recombination rate* van 0,5
Mutation	Insertion met *mutation rate* van 1/ (2 x aantal huidige vrachtwagens)
Parent selection	*Tournament selection with replacement* (*met twee deelnemers*)
Survival selection	*Deterministic Elitist Replacement by* $(\mu + \lambda)$ *selection* (met 75% van de beste)
Population size	100
Number of offspring	100
Initialization	De oplossing van het CSA en de rest *random* toelaatbare oplossingen gegenereerd voor het aanvullen van de populatie
Termination condition	Als het aantal iteratiestappen de 1000 bereikt

Tabel 5: Geselecteerde parameters voor het gebruikte Genetic Algorithm (GA)

6 Resultaten

In dit hoofdstuk worden de resultaten beschreven van de gemaakte planningsmethoden voor het minimaliseren van de ingeplande tijd van de ingezette vrachtwagens. Dit betreft een planning voor de structurele orders van PostNL met ordersoort 60 en 70. In de eerste paragraaf zal de vergelijking met het *Integer Linear Programming (ILP)*, *Concurrent Scheduler Approach (CSA)* en *Genetic Algorithm (GA)* aan de orde komen voor een aangepaste dataset. In de tweede paragraaf worden het CSA en het GA met elkaar vergeleken voor de originele dataset en wordt beschreven waarom de vergelijking met de door PostNL handmatige planning geen reëel beeld geeft.

De drie planningsmethoden zijn elk geprogrammeerd in Python op een laptop met 2.2 GHz 2-Core CPU (i7-2640M) en 8 GB aan geheugen. Voor de implementatie van het ILP is er gebruik gemaakt van twee lineaire programmeertools in Python: PuLP voor het bouwen van het model en CPLEX voor het oplossen ervan. Het gebruikte GA-model hanteert de geselecteerde parameters zoals weergegeven in Tabel 5, daarbij wordt elke run 10 keer gedaan om vervolgens het gemiddelde als resultaat te nemen.

6.1 Model met aangepaste data voor ILP, CSA en GA

Voor de vergelijking van het exacte model (ILP) met de heuristische (CSA) en de meta-heuristische aanpak (GA) is de originele dataset aangepast. Zonder deze aanpassing zou het ILP-model tegen een exponentieel stijgende rekentijd en de maximale geheugencapaciteit van de computer oplopen bij het oplossen van een te groot aantal variabelen. Zoals bij de variabelen wanneer (1) het aantal orders te groot is; (2) er te veel verschillende laad- en loslocaties zijn; (3) de tijdsduren van het laden, rijden en lossen te hoog zijn; (4) de tijdsvensters van het laden en lossen te strikt zijn. (5) Of wanneer er te veel vrachtwagens worden ingezet aan het begin van het model.

		Totale ingeroosterde tijd (in uren)			Aantal vrachtwagens			Rekentijd (in seconden)		
Aantal orders	Procestijd (in uren)	ILP	CSA	GA	ILP	CSA	GA	ILP	CSA	GA
10	18	48	49	48	6	6	6	0,11	0,10	5,84
20	38	96	105	96	12	13	12	2,23	0,25	11,81
30	58	115	129	129	14	15	15	1.704,58	0,45	16,83
40	79	140	180	171	17	21	21	493,28	0,65	21,99
50	101	162	213	213	19	25	25	2.048,14	0,67	27,22

Tabel 6: Resultaten voor het ILP, CSA en het GA voor de aangepaste dataset van 20 juli tot 21 juli 2017. Hierbij betreft elk resultaat van het GA de gemiddelde waarde na 10 keer runnen

Voor het genereren van de aangepaste dataset is een *random* selectie gedaan van orders uit de originele dataset van 20 juli tot 21 juli 2017 met het gekozen tijdsinterval uit hoofdstuk 3. Daarbij worden alle attributen voor een order (zie Tabel 2) behouden, behalve de tijdsvensters voor het laden en lossen. De laad- en losvensters worden hierbij opnieuw gegenereerd. Om deze aanpassing mogelijk te maken worden eerst de tijdsduren voor het laden, lossen en rijden aangepast. Aangezien alle orders gelijke laad- en lostijden hanteren (zie hoofdstuk 3), is er voor de vereenvoudiging gekozen om deze waarden op nul te zetten. De rijtijden daarentegen zijn met een factor 10 verlaagd, zodat wel de verhouding van rijtijden tussen de brieven- en pakketdepots behouden blijft. Om een niet te groot tijdsinterval te genereren voor het laden en lossen en om ervoor te zorgen dat het losmoment kan

aansluiten op het laadmoment met de gekozen rijtijden, zijn de instellingen genomen zoals wordt vermeld in Appendix E.

Om het totaal aantal aangemaakte restricties voor het ILP-model te beperken, zijn de ingezette vrachtwagens van de oplossing van het CSA-model gebruikt. Om het exacte model ook de mogelijkheid te geven om buiten deze vaste startlocaties van vrachtwagens te zoeken, wordt aan elke unieke depotlocatie een extra vrachtwagen toegewezen.

Uiteindelijk resulteerde het doorrekenen van het ILP-model in de resultaten weergeven in Tabel 6. Hierbij is de maximum ordergrootte slechts op 50 gehouden, aangezien het ILP-model na acht uur *runnen* nog geen eindoplossing had gevonden voor 60 orders uit de aangepaste dataset. In Tabel 6 valt op dat het CSA en het GA voor een klein aantal orders dichtbij het ILP-model zit, zowel voor de totale ingeroosterde tijd als voor het aantal vrachtwagens. Het GA heeft zelfs voor een ordergrootte van 10 en 20 de oplossing van het exacte model geëvenaard, terwijl het CSA een vrachtwagen daarvoor extra moet inzetten. Naarmate het aantal orders toeneemt, beginnen het CSA en het GA meer van het exacte model af te wijken. Voor de ordergrootte 30 en 50 zijn de resultaten voor de totale ingeroosterde tijd en het aantal vrachtwagens van het CSA en GA zelfs gelijk aan elkaar. Terwijl voor 40 orders het GA een gelijk aantal vrachtwagens nodig heeft als het CSA, maar acht uur minder tijd ingeroosterd heeft staan. Gekeken naar de duur van de rekentijd steekt het exacte model al snel boven het CSA en het GA uit. Opvallend is de “lage” rekentijd bij de dataset van 40 orders voor het ILP-model, vergeleken met de rekentijd voor de 30 en de 50 orders. Dit wordt veroorzaakt door het feit dat de structurele orders beter op elkaar kunnen aansluiten, wat het inroosteren vergemakkelijkt. Voor de volledigheid is ook de procestijd in Tabel 6 weergegeven. Dit om de verhouding tussen de benodigde tijd voor de orders (procestijd) en de ingeplande tijd (procestijd, wachttijd en duur voor het leeg rijden) van de planningsmodellen aan te geven.

Wel moet hierbij gezegd worden dat de behaalde resultaten van het GA niet gedurende het evolutieproces zijn ontstaan, maar dat dit de oplossingen betreft van de al aan het begin gegenereerde beginpopulatie (één rooster van het CSA en de rest *random* aangemaakt).

6.2 Model met originele data voor CSA en GA

Aangezien het ILP-model te complex is om exact op te kunnen lossen met de originele dataset, worden in deze paragraaf alleen het CSA en het GA met elkaar vergeleken. In Tabel 7 zijn de betreffende resultaten hiervan zichtbaar. Voor de volledigheid is net zoals in Tabel 6 de procestijd erin meegenomen. In Tabel 7 is te zien dat het CSA binnen een redelijke rekentijd van slechts 46 seconden een oplossing voor de gehele dataset heeft gevonden. Hierbij betreft de gevonden oplossing een geldig rooster.

Het GA daarentegen eindigt in een resultaat waar de totale ingeroosterde tijd en het aantal vrachtwagens gelijk is aan of zelfs beter is dan het CSA. Wel moet hierbij gezegd worden dat deze uitkomsten van het GA niet gedurende het evolutieproces zijn ontstaan, maar dat dit de oplossingen zijn van de gegenereerde beginpopulatie (één rooster van het CSA en de rest *random* aangemaakt). Aangezien het niet de oplossingen betreft die tijdens het GA proces ontstaan, zijn de overige ordergroottes in Tabel 7 buiten beschouwing gelaten.

Ook is er een poging gedaan om de door PostNL planning van 20 tot 21 juli 2017 te vergelijken met het resultaat van het CSA. Aangezien de rijtijden anders waren (zie paragraaf 3.3) en de planning van PostNL ook andere ordersoorten bevatte (met geheel andere laad- en loslocaties) zou dit geen reëel beeld geven, daarom is deze planningsvergelijking buiten beschouwing gelaten.

		Totale ingeroosterde tijd (in uren)		Aantal vrachtwagens		Rekentijd (in seconden)	
Aantal orders	Procestijd (in uren)	CSA	GA	CSA	GA	CSA	GA
100	230	792	792	99	99	3,36	175,94
702 (totaal)	1.544	2.824	2798	344	341	45,62	2.462,58

Tabel 7: Resultaten voor het CSA en het GA voor de originele dataset van in totaal 702 structurele orders van 20 juli tot 21 juli 2017. Hierbij betreft elk resultaat van het GA de gemiddelde waarde na 10 keer runnen

7 Conclusies, Restricties & Aanbevelingen

In dit laatste hoofdstuk worden de conclusies, restricties en de aanbevelingen beschreven. Hierbij begint het hoofdstuk met het beschrijven van de conclusies van deze scriptie, met daaropvolgend de restricties en aanbevelingen voor eventueel vervolgonderzoek.

7.1 Conclusies

Om antwoord te kunnen geven op de hoofdvraag zal in deze paragraaf stapsgewijs antwoord gegeven worden op de deelvragen. Deze vragen zullen onderbouwd worden met het literatuuronderzoek (hoofdstuk 2), het data-onderzoek (hoofdstuk 3), de diagnose voor de vorming van het GA (paragraaf 5.2.7), en de resultaten (hoofdstuk 6).

7.1.1 Deelvragen

- ***Hoe zijn de structurele orders over de depots verdeeld?***
  Uit de data (hoofdstuk 3) volgt dat alle werkdagen hetzelfde patroon voor de verdeling van de structurele orders gedurende een dag laten zien. Dit dagelijks patroon laat aan het begin van de avond en rond middernacht de hoogste pieken voor het aantal structurele opdrachten zien (zie Figuur 3a). Ook is hierbij te zien dat de pakketorders de overhand hebben ten opzichte van de categorieën brieven en overig. In Figuur 3b is te zien dat voor de ordersoorten 60 en 70 het grootste gedeelte van de pakketopdrachten in het tijdsvenster van 20:00 - 8:00 uur plaatsvindt. Uit Figuur 4 blijkt dat de werkdagen dinsdag tot en met vrijdag de drukste dagen zijn en de overige werkdagen rustig zijn wat betreft structurele orders. In Figuur 5a is te zien dat de vier pakketdepots (Amersfoort, Dordrecht, 's-Hertogenbosch en Waddinxveen) voor het aantal laad- en losopdrachten ver boven de andere depots uitschieten. Op Dordrecht na vallen de overige drie pakketdepots onder de categorie depot+ (overslaglocatie). Voor pakketdepot Dordrecht is te zien dat de laad- en losopdrachten niet in balans zijn. Dit komt mede doordat er veel losopdrachten in de avond zijn voor Dordrecht (collectie van klanten) en overdag weer meer laadopdrachten op depots (ophalen van emballage). Ook een mogelijke invloed op deze onbalans is dat alle pakketten voor België via Dordrecht gaan.

- ***Welke modellen om het beoogd probleem van PostNL op te lossen zijn bekend in de literatuur?***
  Om na te gaan welke modellen hiervoor bekend zijn in de literatuur is eerst het probleem van deze scriptie geclassificeerd. Uit hoofdstuk 2 volgt dat het beoogd probleem van PostNL het best met de variant *Multi-Depot Pickup and Delivery Problem with Time Windows (MDPDPTW)* vergeleken kan worden. Toch wijkt het beschreven probleem gedeeltelijk hiervan af, omdat het transport van de PostNL orders zich alleen tussen de depots afspeelt. Voor een vrachtwagen dient een van deze depots als zijn hoofddepot. Hierdoor kunnen orders die een vrachtwagenchauffeur uitvoert ook van of naar zijn toegewezen hoofddepot plaatsvinden. Daarnaast wordt een vrachtorder altijd volledig geladen of gelost.

  Aangezien het MDPDPTW onder de complexiteitsgraad van NP-moeilijk valt [12], wil zeggen dat er geen algoritme is om het probleem in polynomiale tijd op te lossen naarmate het probleem groter in omvang wordt. Hierbij wordt de probleemgrootte gedefinieerd als het aantal orders, klanten of locaties waarmee rekening gehouden moet worden. Bij de toepassing van een exacte methode voor *real-world* situaties ligt het aantal variabelen al snel te hoog, waardoor het oplossen van het

probleem al snel te gecompliceerd wordt. Voor het vinden van een oplossing voor grotere probleem instanties binnen een redelijke rekentijd, wordt er vaak naar een heuristiek of meta-heuristiek uitgeweken.

Gegeven de schaarsheid van het aantal *papers* over MDPDPTW in de literatuur en dat het aanpassen van de restricties van het MDPDPTW naar het beoogd transportprobleem van PostNL lastig is, is het model voor PostNL opgebouwd uit de beter gedocumenteerde variant *Vehicle Routing Problem with Time Windows (VRPTW)*. Om inzicht te krijgen in welke restricties nodig zijn en om na te gaan waar de grens ligt om het model van PostNL exact op te lossen, is er gekozen om een *Integer Linear Programming (ILP)* te implementeren. De implementatie van het ILP-model voor dit onderzoek is gebaseerd op de wiskundige formulering gebruikt in Van Barneveld [22] en Neumann [15] voor het inroosteren van opdrachten.

Om alle structurele orders in te kunnen roosteren is de *greedy* heuristiek *Concurrent Scheduler Approach (CSA)* gekozen, aangezien het CSA eenvoudig en snel een toelaatbare oplossing voor het *Vehicle Routing Problem with Time-Windows* [10] vindt. Wel moet het model zo worden uitgebreid dat een depotlocatie zowel een laad- of losplaats voor vracht is en tevens hoofddepot voor de vrachtwagens die daar hun route beginnen of eindigen.

Om te controleren in hoeverre er een betere oplossing gevonden kan worden is er in dit onderzoek ook een meta-heuristische aanpak gedaan. In de literatuur wordt veelal voor het oplossen van varianten van VRP problemen een G*enetic Algorithm (GA)* gebruikt [19]. In [16] wordt een GA toegepast dat zelfs betere oplossingen kan vinden voor enkele instanties uit een tweetal bekende benchmark problemen (zie hoofdstuk 2). Dit komt mede door het gebruik van een nieuwe representatie van een rooster. Ook heeft deze representatie potentie om uitgebreid te worden naar een VRP met tijdsvensters. Daarom is er voor deze scriptie gekozen om als meta-heuristische aanpak het GA te nemen gebaseerd op [16]. Het beoogd model voor PostNL zal hierbij stapsgewijs opgebouwd en onderbouwd worden met de beschreven theorie over evolutionaire algoritmen uit [6].

- ***In hoeverre is het probleem exact op te lossen met een Integer Linear Programming (ILP)* model*?*** Uit paragraaf 6.1 volgt bij het verhogen van de complexiteit van de inputdata voor het ILP-model dat het exacte model al snel tegen een exponentieel stijgende rekentijd en de maximale geheugencapaciteit van de computer oploopt. Dit is het geval wanneer (1) het aantal orders te groot is; (2) er te veel verschillende laad- en loslocaties zijn; (3) de tijdsduren van het laden, rijden en lossen te hoog zijn; (4) de tijdsvensters van het laden en lossen te strikt zijn. (5) Of wanneer er te veel vrachtwagens worden ingezet aan het begin van het model.

  Voor het genereren van resultaten uit het ILP-model is er een aangepaste dataset gemaakt. Deze dataset bestaat uit een *random* selectie van orders uit de originele dataset, waarbij alleen de tijdsvensters voor het laden en lossen opnieuw gegenereerd worden om een zeer klein tijdsinterval te vormen (zie de genomen instellingen vermeld in Appendix E). Daarnaast zijn de laad- en lostijden op nul gehouden en de berekende rijtijden met een factor 10 verlaagd. Ook is om het totaal aantal aangemaakte restricties voor het ILP-model te beperken, de ingezette vrachtwagens van de oplossing van het CSA gebruikt met per uniek depot een toewijzing van een extra vrachtwagen. Voor een klein aantal orders van 50 uit de aangepaste dataset kan het ILP-model een optimale oplossing vinden binnen een redelijke rekentijd (zie Tabel 6). Wel is er een

flinke toename in de rekentijd waarneembaar van nog geen enkele seconde voor 10 orders naar een halfuur voor 50 orders. Als de orders beter op elkaar kunnen aansluiten, zoals bij de ordergrootte van 40 het geval was, wordt het inroosteren aanzienlijk vergemakkelijkt. Bij toename van het aantal orders uit deze zeer gereduceerde dataset blijkt dat voor 60 orders het ILP-model na acht uur *runnen* nog geen eindoplossing gevonden had. Dit betekent dat er naar een andere oplosmethode uitgeweken moet worden om een planning te kunnen genereren voor de originele dataset.

- ***Kan een heuristiek of een meta-heuristiek op basis van het eerder geformuleerde ILP-model en het literatuuronderzoek een oplossing geven binnen een redelijke rekentijd?***
  In Tabel 7 zijn de resultaten zichtbaar voor de originele dataset voor het CSA (*greedy* heuristiek) en het GA (meta-heuristiek). Hierin valt te zien dat het CSA een toelaatbare oplossing kan vinden binnen een rekentijd van slechts 46 seconden voor de gehele dataset. Wel kan het zijn dat de gevonden oplossing van het CSA een suboptimale oplossing betreft, omdat deze heuristiek tijdens het genereren van het rooster te snel een nieuwe vrachtwagen inzet.

  Het GA daarentegen eindigt in een resultaat waar de totale ingeroosterde tijd en het aantal vrachtwagens gelijk is aan of zelfs beter is dan het CSA (zie Tabel 7). Wel moet hierbij gezegd worden dat deze uitkomsten van het GA niet gedurende het evolutieproces zijn ontstaan, maar dat dit de oplossingen zijn van de gegenereerde beginpopulatie (één rooster van het CSA en de rest random aangemaakt). Tijdens het vormen van het GA (zie paragraaf 5.2.7) is het vinden van oplossingen waar alle orders aan routes worden gekoppeld niet het tijdrovende gedeelte van het GA – het optimaliseren van orders naar een geldig rooster met een zo min mogelijke ingeroosterde tijd wel. Ondanks het vele testen en bijstellen ("*trial-and-error*") van de stappen in het GA en de daarvoor gebruikte parameters blijven er in de *recombination* fase en de *mutation* fase voornamelijk ontoelaatbare oplossingen (ongeldige roosters) ontstaan. Bij gebruik van alleen de variatie operator *mutation* worden er vaker toelaatbare oplossingen gecreëerd. Wel zijn de beperkt aantal gecreëerde oplossingen bij zowel *mutation* met of zonder *recombination* niet beter dan de oplossingen van de beginpopulatie (één rooster van het CSA en de rest random aangemaakt). Hierdoor blijft na een aantal opeenvolgende generaties de diversiteit in de populatie sterk afnemen, waarbij de gehele populatie na een korte tijd *runnen* naar een lokaal optimum leidt (*premature convergence*). Het is zelfs zo dat de gehele populatie van het GA gelijk is aan één van de al gegenereerde oplossingen uit de beginpopulatie.

- ***Indien de nieuwe oplossing een lagere totale ingeplande tijd heeft, welke vrachtwagens moeten er afgeschaald/opgeschaald worden t.o.v. de vorige planning?***
  Helaas kan op deze deelvraag geen antwoord gegeven worden, aangezien de rijtijden anders waren (zie paragraaf 3.3) en de door PostNL handmatige planning in diezelfde periode ook andere ordersoorten bevatten (met geheel andere laad- en loslocaties). Daarom is deze planningsvergelijking buiten beschouwing gelaten.

7.1.2 Hoofdvraag

Om als PostNL op de veranderende postmarkt in te spelen, is het van belang om na te gaan waar de efficiëntie in het transportproces verbeterd kan worden. Deze scriptie richt zich op de planningsfase van groottransport tussen de brieven- en pakketdepots van PostNL. Het doel van dit onderzoek is om na te gaan of de ritten (diensten bestaande uit de structurele orders van PostNL met ordersoort 60 en 70) van de vrachtwagens efficiënter ingepland kunnen worden door gebruik te maken van wiskundige modellen.

Gebaseerd op de aannames gemaakt in paragraaf 3.3 en de conclusies uit de voorgaande paragraaf kan de volgende hoofdvraag beantwoord worden:

"Hoeveel efficiënter kan PostNL haar planning voor de structurele orders maken door gebruik te maken van wiskundige modellen?"

Met het gebruik van de *Concurrent Scheduler Approach (CSA)* zou PostNL snel een toelaatbare oplossing krijgen voor de structurele orders met ordersoort 60 en 70. Wel kan het zijn dat de gevonden oplossing van het CSA een suboptimale oplossing betreft, omdat deze heuristiek tijdens het genereren van het rooster te snel een nieuwe vrachtwagen inzet. Het G*enetic Algorithm (GA)* daarentegen loopt op tegen de ontoelaatbare oplossingen die ontstaan doordat de tijdsvensters van het laden en lossen en arbeidstijden van een vrachtwagenchauffeur overschreden worden. Door meer "*trial-and-error*" voor de stappen in het GA en de daarvoor gebruikte parameters heeft het GA potentie om een goede oplossing te vinden, maar vervolgonderzoek zal daarvoor nodig zijn. Ondanks dat het exacte *Integer Linear Programming (ILP)* model geen oplossing vindt voor grote instanties van het probleem kan dit model een goede opzet geven om een planningsmodel op te bouwen en te testen voor een klein schaalmodel. In Tabel 8 is een overzicht gegeven van de voor- en nadelen van de drie gebruikte planningsmethoden.

Planningsmethode	Voordelen	Nadelen
ILP	• Optimale oplossing • Kan opzet geven voor het bouwen en testen van een ander planningsmodel	• Beperkt aantal orders • Vrachtwagens vooraf instellen
CSA	• Snel toelaatbare oplossing	• Suboptimale oplossing
GA	• Potentie op goede oplossing (vervolgonderzoek nodig)	• "*Trial-and-error*" voor de stappen in het GA en de daarvoor gebruikte parameters • Ontoelaatbare oplossingen tijdens zoeken

Tabel 8: Overzicht van de voor- en nadelen van de gebruikte planningsmethoden

7.2 Restricties

In paragraaf 3.3 zijn enkele aannames gemaakt om een algemeen model te vormen voor de drie gemaakte planningsmethoden. Zo wordt er altijd vanuit gegaan dat alle vrachtwagens leeg starten en leeg (of vol) eindigen bij hun toegewezen depot. Daarnaast is er een vaste tijd voor het laden en lossen gehanteerd van elk 25 minuten. In de praktijk zou deze duur mogelijk ook hoger uit kunnen pakken, omdat er soms te weinig mensen in het depot beschikbaar zijn om de chauffeur te helpen. Het kan zelfs zo zijn dat een vrachtwagenchauffeur er alleen voor staat om de vracht te moeten laden of lossen. Dit heeft als gevolg dat de rest van de planning in de war zou komen.

Aangezien de focus van deze scriptie alleen lag op de structurele orders met ordersoort 60 en 70 kon de gegenereerde planning niet met de door PostNL handmatige planning vergeleken worden. Dit kwam omdat de PostNL planning in diezelfde periode ook andere ordersoorten bevatte (met geheel andere laad- en loslocaties). Daarnaast kwamen zowel in de ontvangen dataset als de door PostNL gemaakte planning niet alle verbindingen tussen de depots voor. Vanwege het ontbreken van deze gegevens zijn de rijtijden tussen de verschillende depots berekend (zie paragraaf 3.3).

7.3 Aanbevelingen

Als vervolgonderzoek kan gekeken worden naar het verbeteren van het *Genetic Algorithm (GA)*, aangezien deze oplosmethode zeker potentie heeft om een goede oplossing te vinden. Bijvoorbeeld daar waar *mutation* plaatsvindt kan door sturing slechte (weinig aansluitende orders) roosters worden aangepast, zodanig dat er een toelaatbaar en goed rooster behouden blijft.

Ook interessant is het om een gevoeligheidsanalyse te doen als de laad- en losvensters aangepast worden door het vergroten van het tijdsvenster met bijvoorbeeld tien minuten. Naast de tijdsrestrictie zou er ook onderzoek gedaan kunnen worden naar het scenario waarin de vrachtwagenchauffeur niet terug hoeft te keren naar het startdepot. Een ander vervolgonderzoek zou het maken van regio-afhankelijke planningen kunnen zijn, in plaats van het maken van een planningsmodel voor heel Nederland.

Appendices

A Lijst met afkortingen

ILP	*Integer Linear Programming*
COP	*Constrained Optimization Problem*
CSA	*Concurrent Scheduler Approach*
GA	*Genetic Algorithm*
KO	Kwantitatieve Ondersteuning
MDPDPTW	*Multi Depot Pickup and Delivery Problem with Time Windows*
MDVRPTW	*Vehicle Routing Problem with Multiple-Depots with Time Windows*
OS	Order Soort
PSO	*Particle Swarm Optimization*
RPDPTW	*Rich Pickup and Delivery Problem with Time Windows*
SCB	Sorteercentra voor Brieven
SCP	Sorteercentra voor Pakketten
VRP	*Vehicle Routing Problem*
VRPTW	*Vehicle Routing Problem with Time Windows*

Tabel 9: Lijst met gebruikte afkortingen

B Lijst met definities

Arbeidstijd	Alle tijd tussen twee dagelijkse/wekelijkse rusttijden (m.a.w. alle gewerkte uren)
Incidenteel	Eenmalige transportopdracht uiterlijk 17.00 uur dag vóór uitvoering aangemeld vrijdag uiterlijk 18.00 uur voor maandag
Interritten	De ritten in Nederland die de vrachtwagens tussen de sorteercentra van brieven en pakketten afleggen
Laadvenster	Het venster waarbinnen de auto moet aankomen, het laadproces uit moet voeren en moet vertrekken
Losvenster	Het venster waarbinnen de auto moet aankomen en moet beginnen met lossen
Pauze	Rust tussen twee rijtijden/arbeidstijden (m.a.w. rust tijdens de dienst)
Rijtijd	Alle tijd dat er daadwerkelijk wordt gereden (inclusief file e.d.)
Spoed	Transportopdracht later dan 17.00 uur dag vóór uitvoering aangemeld
Structureel	Minimaal één dag per week met dezelfde laad- en losadressen, tijdsvensters en verwachte belading over een periode van minimaal 3 weken

Tabel 10: Lijst met definities

C Adressen van de PostNL brieven- en pakketdepots

Depot	Plaats	Postcode	Straat & Huisnummer
1	Amsterdam	1046 BP	Australiehavenweg 100
2	Den Haag	2491 AN	Loire 1
3	Rotterdam	3062 CZ	Terbregseweg 300
4	Nieuwegein	3439 NT	Grote Wade 80
5	‘s-Hertogenbosch	5215 ME	Steenbok 2
6	Zwolle	8013 NS	Anthony Fokkerstraat 4

Tabel 11a: Adressen van de zes PostNL brievendepots

Depot	Plaats	Postcode	Straat & Huisnummer
1	Amersfoort	3815 KP	De Windturbine 8
2	Born	6121 RE	Holtum-Noordweg 107
3	Breda	4824 AP	Steltbeemd 1
4	Den Hoorn	2635 DK	Heernesse 10
5	Dordrecht	3316 GZ	Pieter Zeemanweg 104
6	Elst	6662 NG	Nijverheidsweg 2 D
7	Goes	4462 HB	Columbusweg 62
8	Halfweg	1165 AA	Linieweg 10
9	Hengelo	7554 RR	Zuidelijke Havenweg 1
10	Kolham	9615 TP	W.J. Ockelslaan 2
11	Leeuwaarden	8912 BL	Brailleweg 3
12	Opmeer	1716 KJ	De Veken 332
13	Ridderkerk	2988 CK	Schaapherderweg 21
14	Sassenheim	2171 TX	Einsteinpark 2
15	‘s-Hertogenbosch	5222 AL	Ketelaarskampweg 4
16	Son	5692 CK	Ekkersrijt 3402
17	Utrecht	3542 AR	Sophialaan 7 A
18	Waddinxveen	2742 RJ	Overslagweg 2
19	Zwolle	8013 RJ	Nipkowstraat 1

Tabel 11b: Adressen van de negentien PostNL pakketdepots

D Datavoorbeeld selectie uit werkweek 29 2017

	A	B	C	D	E	F	G	H	I
1	**Ordernr**	**Opdrachtdatum**	**Laadcode**	**LaadtijdBegin**	**LaadtijdEind**	**Loscode**	**LostijdBegin**	**LostijdEind**	**SoortOrder**
2	27541627	17-7-2017	1046BP100S	23:50	00:15	3439NT80S	23:50	01:15	60
3	27541628	18-7-2017	1046BP100S	23:50	00:15	3439NT80S	23:50	01:15	60
4	27541629	19-7-2017	1046BP100S	23:50	00:15	3439NT80S	23:50	01:15	60
5	27541630	20-7-2017	1046BP100S	23:50	00:15	3439NT80S	23:50	01:15	60
6	27541631	21-7-2017	1046BP100S	23:50	00:15	3439NT80S	23:50	01:15	60
7	27541632	17-7-2017	1046BP100S	23:20	23:45	3439NT80S	23:20	00:45	60
8	27541633	18-7-2017	1046BP100S	23:20	23:45	3439NT80S	23:20	00:45	60
9	27541634	19-7-2017	1046BP100S	23:20	23:45	3439NT80S	23:20	00:45	60
10	27541635	20-7-2017	1046BP100S	23:20	23:45	3439NT80S	23:20	00:45	60

Figuur 14: Voorbeeld uit de PostNL dataset van werkweek 29 met de geselecteerde variabelen

E Instellingen voor de tijdsvensters van de aangepaste orderdata

[A] LaadtijdBegin	[B] LaadtijdEind	[C] LostijdBegin	[D] LostijdEind
Random (0;5)	A + *Random* (1;2)	A	B + *Random* (3)

Tabel 12: Instellingen voor het genereren van de tijdsvensters voor de aangepaste orderdata

Bibliografie

[1] Autoriteit Consument & Markt (ACM). (2017). Jaaroverzicht Post- en Pakkettenmonitor 2016. Geraadpleegd van https://www.acm.nl/sites/default/files/old_publication/publicaties/17545_Post-en-pakkettenmonitor-2016-08-08-2017.pdf

[2] Capacitated VRP instances. (2013). Geraadpleegd op 7 februari, 2018, van neo.lcc.uma: http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/

[3] Dantzig, G., & Ramser, J. (1959). The truck dispatching problem. Management Science, 6, 80-91.

[4] Eiben, A.E, & Schippers, A (1998). On evolutionary exploration and exploitation. Fundamenta Informaticae, 35(1-4), 35-50.

[5] Eiben, A.E., & Smith, J.E. (2015). Introduction to Evolutionary Computing (2de ed.). Springer.

[6] Flisberg, P., Lidén, B., & Rönnqvist, M. (2009). A hybrid method based on linear programming and tabu search for routing of logging trucks. Computers & Operations Research, 36(4), 1122-1144.

[7] Goela, A., & Gruhn, V. (2008). A general vehicle routing problem. European Journal of Operational Research, 191(3), 650-660.

[8] Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

[9] Haksever, C., Render, B., Russell, R., & Murdick, R. (2000). Service Management and Operations (2de ed.). Prentice Hall: Upper Saddle River, 476-497.

[10] Karakatič, S., en Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. Applied Soft Computing, 27, 519-532.

[11] Lenstra, J.K. & Rinnooy Kan, H.G (1981). Complexity of vehicle routing and schedulingproblems. Networks, 11, 221-227.

[12] Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimisation problems. Evolutionary Computation, 4(1), 1-32.

[13] Montoya-Torres J.R., Franco J.L., Isaza S.N. e.a. (2015). A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering, 79, 115-129.

[14] Neumann, K., Schwindt, C., & Zimmermann, J. (2003). Project scheduling with time windows and scare resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. Berlin: Springer.

[15] Pereira, F.B., Tavares, J., Machado, P., & Costa, E. (2002). GVR: A New Genetic Representation for the Vehicle Routing Problem. Springer-Verlag Berlin Heidelberg.

[16] Pereira, F.B., & Tavares, J. (2009). Bio-inspired Algorithms for the Vehicle Routing Problem, Studies in Computational Intelligence. Springer Berlin / Heidelberg, 161.

[17] Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. Computers & Operations Research, 34(8), 2403-2435.

[18] Potvin, J. (2009). State-of-the Art Review: Evolutionary Algorithms for Vehicle Routing. INFORMS Journal on Computing 21(4), 518-548.

[19] Sombunthama, P., & Kachitvichyanukulb, V. (2010). Multi-depot vehicle routing problem with pickup and delivery requests. AIP Conference Proceedings of the International MultiConference of Engineers and Computer Scientists, 1285, 71-85.

[20] Troya, J.M. (2002). Improving flexibility and efficiency by adding parallelism to genetic algorithms. Statics and Computing, 12(2), 91-114.

[21] Van Barneveld, T.C. (2012). *Heuristic Methods for Makespan Minimization in Project Scheduling* (master scriptie). Geraadpleegd van https://www.math.leidenuniv.nl/nl/theses/331/

[22] Weise, T., Podlich, A., & Gorldt, C. (2010). Solving real-world vehicle routing problems with evolutionary algorithms. In R. Chiong, & S. Dhakal (Eds.), Natural intelligence for scheduling, planning and packing problems, 29-53.