UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur

Masterthesis

Scattered storage in hedendaagse distributiecentra: locatiebeslissing, locatie-selectie en routering

Aïcha Leroy

Scriptie ingediend tot het behalen van de graad van master handelsingenieur, afstudeerrichting operationeel management en logistiek

PROMOTOR :

Prof. dr. Kris BRAEKERS

2020
2021

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur

Masterthesis

Scattered storage in hedendaagse distributiecentra: locatiebeslissing, locatie-selectie en routering

Aïcha Leroy

Scriptie ingediend tot het behalen van de graad van master handelsingenieur, afstudeerrichting operationeel management en logistiek

PROMOTOR :

Prof. dr. Kris BRAEKERS

Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op het schrijf- en verwerkingsproces, de onderzoekshandelingen en de onderzoeksresultaten die aan de basis liggen van dit werkstuk.

Woord vooraf

In deze masterproef wordt een centrale onderzoeksvraag beantwoord omtrent de toepassing van de opslagstrategie scattered storage in hedendaagse distributiecentra. Deze studie werd geschreven ter afsluiting van mijn studies als Handelsingenieur – Operationeel Management en Logistiek. Het was een uitdagend project tijdens de coronaperiode waarbij ik vele uren achter mijn computerscherm spendeerde. Toch heb ik altijd met veel plezier eraan gewerkt. Dat enthousiasme had ik niet kunnen bewaren zonder de hulp van een aantal mensen.

Eerst en vooral wil ik mijn promotor bedanken voor de vele feedbackmomenten en om zo vaak mijn teksten opnieuw te willen nalezen. Na een feedbackmoment wist ik concreet wat de volgende stappen waren en dat gaf me altijd motivatie om er opnieuw in te vliegen. Daarnaast wil ik mijn ouders bedanken om mij te steunen tijdens mijn vijf opleidingsjaren en mijn vriend om altijd mijn schoolwerk te willen nakijken alvorens ik het instuurde.

Samenvatting

1 Inleiding

Business-to-consumer e-commerce staat bekend als de elektronische vorm van handel via het internet tussen bedrijven en consumenten. De toename van persoonlijke computers, tablets en smartphones heeft het doelpubliek van deze handelswijze aanzienlijk vergroot. Zo zouden in 2020 meer dan 7 op 10 Belgen een online aankoop gedaan hebben. Door het gemak van e-commerce is een trend ontstaan om steeds frequenter, maar in kleinere hoeveelheden te bestellen. Deze stijging in het aantal bestellingen legt druk op het leveringssysteem. Daarom zullen magazijnen (waar activiteiten worden uitgevoerd zoals het ontvangen van goederen, het stockeren van goederen, de orderverzameling en de verzending van orders) genoodzaakt zijn om operationele kosten te minimaliseren. Scattered storage optimaliseert het order picking proces[1], dat gewoonlijk de grootste operationele kost is in een magazijn. Eenzelfde product (later vernoemd als Stock Keeping Unit (SKU)) wordt bij deze opslagmethode als afzonderlijke artikels over de reklocaties verspreid en bevindt zich daardoor op meerdere plaatsen in het magazijn. Dat is in tegenstelling tot traditionele opslagmethoden waarbij elk product slechts in een beperkt gebied van het magazijn beschikbaar is. Kleine orders worden significant sneller gepickt en de klant ontvangt vlugger zijn/haar bestelling.

Nog maar weinig onderzoek werd uitgevoerd naar scattered storage en daarom werd de volgende onderzoeksvraag opgesteld: *Hoe kan in een scattered storage picker-to-parts*[2] *magazijn van een business-to-consumer e-commerce bedrijf het pickproces optimaal georganiseerd worden?* Deze vraag wordt beantwoord d.m.v. een literatuurstudie om bekend te worden met het onderwerp gevolgd door een praktijkstudie om verschillende scenario's te testen met realistische data.

2 Literatuurstudie

In het eerste deel van de literatuurstudie worden de te nemen beslissingen in een magazijn besproken in twee secties. De eerste sectie bestaat uit de beslissingen gerelateerd aan de lay-out van het magazijn die opgedeeld zijn in vier delen: de structuur, de afmetingen, de departementen en de apparatuur. De tweede sectie betreft de beslissingen over de operationele werking van het magazijn die onderverdeeld zijn in drie sub-secties: ontvangst- en verzendactiviteiten, de opslagafdeling en het order picking proces.

Het tweede deel van de literatuurstudie beschrijft het huidige onderzoek over scattered storage. De eerste paper gaat over het stockeringsprobleem. Dat probleem tracht de SKUs in het magazijn optimaal te spreiden en werd door de auteurs opgelost d.m.v. een heuristiek. Deze oplossingsmethode bleek over het algemeen kortere pickroutes te vinden dan een willekeurige opslagmethode, waarbij de SKUs een willekeurige vrije locatie toegewezen krijgen. De tweede en de derde paper gaan beiden over het tweedelige picker-routing probleem. Eerst moet worden bepaald van welke opslaglocatie een SKU gepickt zal worden en vervolgens moet een route tussen de gekozen

[1] Tijdens het order picking proces worden goederen uit een magazijn verzameld op aanvraag van een klant.

[2] Picker-to-parts is een verzamelsysteem waarbij de pickers zich doorheen het magazijn verplaatsen naar de opslaglocaties van de te picken SKUs.

opslaglocaties worden samengesteld. In de tweede paper werden drie verschillende prioriteitsheuristieken ontwikkeld die het opslaglocatie-selectie probleem oplossen, namelijk SinglePosition, MinMin en MinMax. De derde paper stelde nog twee andere oplossingsmethoden voor hetzelfde probleem voor. De eerste methode was een nearest-neighbour search en de tweede een pool-based constructie heuristiek.

In het derde deel van de literatuurstudie werd op zoek gegaan naar toepassingen die gelijkaardig zijn aan het picker-routing probleem. De eerste toepassing is het General Traveling Salesman Problem (GTSP). Hierbij worden klanten opgedeeld in clusters en het voertuig moet elke cluster precies één keer bezoeken. De tweede toepassing is het General Vehicle Routing Problem (GVRP). Daarbij is het doel om de optimale route te vinden voor elk van de voertuigen uit het wagenpark met vertrek- en eindpunt in het depot, die precies één klant uit elke groep bezoekt en dat rekening houdend met de capaciteitsbeperkingen van de wagens. In beide methoden is zo een cluster vergelijkbaar met alle opslaglocaties die een specifieke SKU bevatten. Een picklijst kan dan voltooid worden door voor elke te picken SKU de bijbehorende cluster van opslaglocaties te bezoeken.

3 Praktijkstudie

Tijdens de praktijkstudie werden drie doelen onderzocht die elk gebaseerd zijn op het picker-routing probleem in een scattered storage magazijn. In totaal werden zes scenario's uitgevoerd met een verkregen realistische dataset. Daarvoor werden telkens de route-tijden van elke order over alle picklijsten heen opgeteld om de scenario's met elkaar te kunnen vergelijken.

Het eerste doel was om de prioriteitsheuristieken uit de literatuur te testen op een nieuwe dataset. Uit de resulaten van dat eerste scenario bleek dat de MinMin heuristiek de beste oplossingsmethode is en dat de SinglePosition heuristiek soms tot beduidend slechtere oplossingen leidt. Deze resultaten stemmen overeen met die uit de paper, behalve dat daar de MinMax methode het vaakst tot slechtere oplossingen leidde. Daarnaast werd een totale route-tijd van 1.623 minuten en 22 seconden voor het eerste scenario gevonden.

Het tweede t.e.m. het vierde scenario testten het tweede doel van de praktijkstudie, namelijk of batching een verbetering zou teweegbrengen in de route-tijden en welke batching methode best gebruikt kan worden. Eerst werd een First-In-First-Out (FIFO) batching methode met een batchgrootte van twee orders getest. Dat wil zeggen dat orders per twee in een batch gegroepeerd worden o.b.v. de volgorde op de picklijst. Een daling van 507 minuten en 25 seconden ofwel 31,26% werd waargenomen t.o.v. het eerste scenario. Het derde en vierde scenario, SinglePosition batching met batchgrootte twee, testten een batching methode uit gebaseerd op één van de prioriteitsheuristieken met telkens een verschillend startpunt. Indien als startpunt de SKU met het minste aantal opslaglocaties werd gekozen, dan werd een route-tijd van 1.098 minuten en 18 seconden gevonden en indien het startpunt de SKU was met het meeste aantal opslaglocaties, dan werd een route-tijd van 1.087 minuten en 42 seconden gevonden. Bijgevolg is SinglePosition batching met als startpunt de SKU met het meeste aantal opslaglocaties de best gevonden batching methode voor de huidige dataset, maar het verschil met SinglePosition batching met het andere startpunt is slechts klein.

Het derde doel van de praktijkstudie was om de opslaglocatie-selectie in het picker-routing probleem op te lossen d.m.v. een parallelle/gelijktijdige beslissing voor alle te picken SKUs. Dat is in tegenstelling tot de sequentiële SKU per SKU beslissing van de prioriteitsheuristieken. Twee formuleringen werden opgesteld als vijfde en zesde scenario. De eerste formulering trachtte een route te maken tussen de geselecteerde opslaglocaties, maar had als nadeel dat subtours zich konden vormen en daardoor werd de afgelegde afstand onderschat. De tweede formulering daarentegen probeert geen route te maken, maar telt alle mogelijke paden op tussen de geslecteerde opslaglocaties. Daardoor werd de route-tijd daar overschat. Na de opslaglocatie-selectie via de formuleringen werd een route gevormd tussen de locaties. De onderschattende methode vond een route-tijd van 1.708 minuten en 40 seconden en de overschattende methode vond een route-tijd van 1.731 minuten en 35 seconden. Het verschil tussen de methoden is bijgevolg klein, maar ze presteren beide slechter dan het eerste scenario. Zo vond de onderschattende methode een route-tijd die 5,25% hoger lag dan het eerste scenario en de overschattende methode vond een stijging van 6,67%. Verrasingwekkend lijkt de conclusie hier dat voor de huidige dataset de gelijktijdige beslissing geen verbetering met zich meebrengt in functie van de route-tijd.

4 Conclusies en kritische bemerkingen

Uit deze masterproef kan geconcludeerd worden dat bij het oplossen van de prioriteitsheuristieken met een nieuwe dataset gelijkaardige resultaten als in de literatuur worden gevonden. Indien batching methoden worden toegepast, dan wordt de algemene route-tijd vermindert, maar het is bij SinglePosition batching dat de grootste vermindering wordt gevonden. Daarnaast lijkt een gelijktijdige beslissing voor het opslaglocatie-selectie probleem toch geen verbetering in de route-tijden teweeg te brengen. Een beperking van de praktijkstudie is dat in totaal 180 picklijsten met telkens zes orders gebruikt werden om de scenario's te testen. De hoeveelheid data is bijgevolg eerder beperkt, waardoor de conclusies met voorzichtheid getrokken moeten worden.

Een kritische bemerking is dat het enkel interessant is om scattered storage toe te passen indien het voordeel van verkorte picktijden opweegt tegen de verhoogde inspanning om de rekken bij te vullen. Opslaglocaties komen namelijk sneller vrij doordat deze slechts gevuld zijn met een kleine hoeveelheid SKUs. Daarnaast is het realistisch dat een magazijn niet enkel e-commerce leveringen moet voltooien, maar eveneens leveringen in grotere hoeveelheden aan fysieke winkels. Daarom is het pas interessant om scattered storage toe te passen indien minimum 33% van de bestellingen via het internet worden geplaatst.

Inhoudsopgave

1 INLEIDING **9**

1.1 PRAKTIJKRELEVANTIE 9

1.2 PROBLEEMSTELLING 13

1.3 METHODOLOGIE 14

1.3.1 Literatuurstudie *14*

1.3.2 Praktijkstudie *16*

2 LITERATUURSTUDIE **19**

2.1 BESLISSINGEN IN EEN MAGAZIJN 19

2.1.1 Lay-out van het magazijn *19*

2.1.2 De operationele werking van het magazijn *23*

2.2 SCATTERED STORAGE 29

2.2.1 Lay-out van het magazijn *30*

2.2.2 Stockering *30*

2.2.3 Picker-routing *34*

2.3 GELIJKAARDIGE TOEPASSINGEN 41

2.3.1 Generalized traveling salesman problem (GTSP) *42*

2.3.2 Generalized vehicle routing problem (GVRP) *43*

3 PRAKTIJKSTUDIE **47**

3.1 HET VOORBEELD MAGAZIJN 47

3.2 HET OPSTELLEN VAN VOORBEELDDATA 48

3.3 TESTEN VAN DE PRIORITEITSHEURISTIEKEN OP NIEUWE DATA 50

3.4 OPLOSSINGSPRESTATIES NA BATCHING 53

3.4.1 FIFO batching *54*

3.4.2 SinglePosition batching met de SKU met de minste opslaglocaties als startpunt *56*

3.4.3 SinglePosition batching met de SKU met de meeste opslaglocaties als startpunt *59*

3.4.4 Vergelijking van de batching methoden *61*

3.5 OPLOSSINGSPRESTATIES NA EEN GELIJKTIJDIGE BESLISSINGING VAN OPSLAGLOCATIES 62

3.5.1 Algemene onderschatting *63*

3.5.2 Algemene overschatting *64*

3.5.3 Vergelijking van de parallelle selectie methoden *65*

4 CONCLUSIE **69**

5 BIBLIOGRAFIE **73**

6 BIJLAGEN **79**

6.1 DE VOORBEELD MAGAZIJNEN 79

6.2 DE OPLOSSINGSPRESTATIES PER PICKLIJST-GROEP 82

6.2.1 De prioriteitsheuristieken met nieuwe data *82*

6.2.2 FIFO-batching *83*

6.2.3 SinglePosition batching *84*

6.2.4 SinglePosition batching met een alternatieve eerste SKU *85*

6.2.5 De parallelle opslaglocatie-selectie met de onderschattende methode *86*

6.2.6 De parallelle opslaglocatie-selectie met de overschattende methode *86*

6.3 DE LINGO CODE VAN DE PARALLELLE OPSLAGLOCATIE-SELECTIE FORMULERING 87

6.3.1 De Lingo code van de onderschattende methode *87*

6.3.2 De Lingo code van de overschattende methode *88*

1 Inleiding

In de inleiding worden drie delen besproken, namelijk de praktijkrelevantie, de probleemstelling en de methodologie. Het deel praktijkrelevantie zal de context van het onderzoeksonderwerp beschrijven. Vervolgens wordt in de probleemstelling de onderzoeksvraag gedefinieerd en de bijbehorende deelvragen verder verklaard. Ten slotte wordt in de methodologie, volgens de structuur van de masterproef, de aanpak van elk deel in detail besproken.

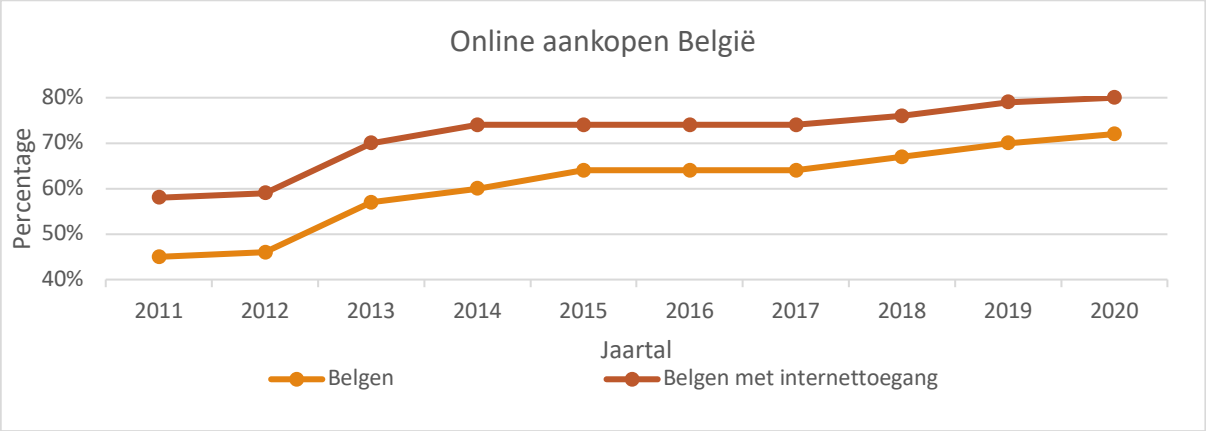
1.1 Praktijkrelevantie

De term supply chain management werd in 1982 bedacht door Keith Oliver. Zijn visie was om de functionele silo's, zoals Accounting, Marketing en Finance, binnen bedrijven te verbreken en zo alle departementen te laten samenwerken (Russell, 2011). Deze departementen kunnen zich zowel binnen eenzelfde bedrijf als over verschillende bedrijven bevinden. Een voorbeeld van departementen verspreid over verschillende bedrijven is een bedrijf dat de verpakking van hun product uitbesteedt aan een ander bedrijf, maar alle andere activiteiten (behorend tot departementen) zelf uitvoert (Van Dale Uitgevers, 2020). Supply chain management wordt aldus als volgt gedefinieerd: een samenwerking tussen opeenvolgende bedrijven of departementen binnen eenzelfde bedrijf om een product of dienst zo efficiënt mogelijk op de markt te brengen (Russell, 2011). Een voorbeeld van supply chain management is de samenwerking tussen magazijnen en leveringsdiensten, waarbij het doel is een product tot bij de klant te brengen. Een sector waarbij deze samenwerking cruciaal is, is de e-commerce sector. In deze sector wordt de supply chain enorm onder druk gezet om de producten zo snel mogelijk op de eindbestemming te brengen.

E-commerce, ofwel electronic commerce, staat bekend als de elektronische vorm van handel drijven via het internet (Van Dale Uitgevers, 2020). Michael Aldrich, de bedenker van e-commerce, had origineel het idee om bejaarden van huis uit hun wekelijkse inkopen te laten doen. Dat was mogelijk via een scherm op de televisie die in 1979 al te vinden was in de meeste huishoudens. Het idee om via het internet met andere bedrijven handel te drijven werd opgepikt en zo ontstond het online B2B (business-to-business) netwerk. In 1981 begonnen de eerste bedrijven zich eveneens te richten op thuisaankopen voor consumenten (B2C, oftewel business-to-consumer). Vervolgens, door het ontstaan en de toename van persoonlijke computers, vergrootte het doelpubliek van online aankopen en ontstonden alsook de terminologieën: e-commerce, e-shopping en e-business. Technologische innovaties, zoals tablets en smartphones, spelen een grote rol in het steeds toenemende aandeel online aankopen (Aldrich, 2011).

In een fysieke winkel vindt de communicatie tussen een bedrijf en de klant plaats via een medewerker in de winkel. Dat is niet meer mogelijk in B2C e-commerce winkels. Deze vermindering in communicatie blijkt echter geen negatieve invloed te hebben op de klantervaring (Kumar & Petersen, 2006). Tevens werd in 2003 door Rust R. en Kannan P. al aangegeven dat het oprichten van een online winkel naast een fysieke winkel noodzakelijk zou zijn om competitief te blijven op vlak van klantervaring (Rust & Kannan, 2003). Dat wordt bevestigd door het stijgend aantal personen

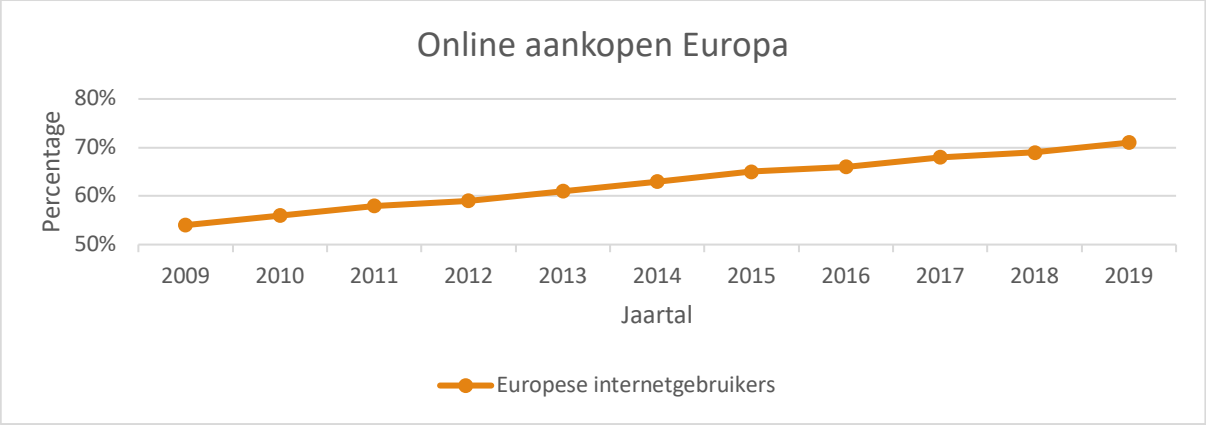
dat online aankoopt. In een recent onderzoek uitgevoerd in België door Comeos[5] werd namelijk bevonden dat meer dan 7 op 10 Belgen in 2020 online aankopen gedaan hebben. Dat loopt zelfs op tot 80% onder de Belgische bevolking die over internettoegang beschikt. Deze percentages zijn mogelijks beïnvloedt door de corona epidemie, maar een grafiek met gegevens van dezelfde studie over tien jaar toont toch een duidelijke groei aan van de populariteit van online aankopen bij Belgen. Tussen 2011 en 2020 is een stijging zichtbaar van 27% onder de Belgische bevolking en een stijging van 22% onder de Belgische bevolking met internettoegang (zie 'Figuur 1'). Een verdere stijging in 2021 wordt verwacht, doordat 61% van de consumenten die nog geen online aankoop heeft gedaan, aangeeft dat volgend jaar wel te willen doen. Indien van de 1.681 ondervraagden uit het onderzoek van Comeos enkel de deelnemers worden geselecteerd die in de laatste twaalf maanden een online aankoop hebben gedaan, dan geeft 56% aan minstens éénmaal per maand online te bestellen. Van dezelfde selectie doet zelfs één op vijf personen wekelijks een online aankoop (Comeos, 2020).



Figuur 1: Percentage van de Belgische bevolking (met internettoegang) die online aankopen doet (Comeos, 2020).

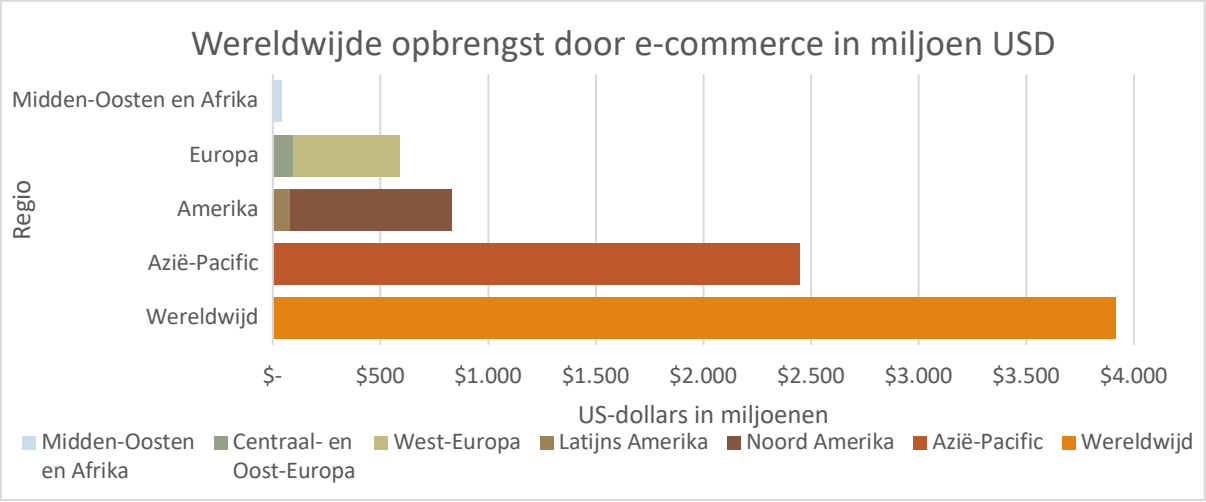
Op Europees vlak stijgt het aantal internetgebruikers en het aantal mensen die over het internet aankopen doen evenzeer (Lone et al., 2019). Van de Europese internetgebruikers heeft meer dan 70% in 2019 een online aankoop gedaan (zie 'Figuur 2') (Eurostat, 2020). In 2017 werd in een onderzoek door mastercard bevonden dat één vierde van de Europese internetgebruikers één keer per week online winkelt. Dat percentage stijgt naar meer dan 60% indien het gaat over minstens één keer per maand online aankopen en zelfs tot 90% bij minstens één keer per jaar online aankopen (mastercard, 2017).

[5] Comeos is de Belgische federatie voor handel en diensten, die jaarlijks een studie uitvoert over e-commerce in België.



Figuur 2: Percentage van de Europese inwoners met internettoegang die online heeft aangekocht (Eurostat, 2020).

Op wereldwijd vlak vertegenwoordigde e-commerce in 2019 al 14,1% van alle retail verkopen (Statista, 2020) en voor 2020 wordt verwacht dat e-commerce bijna 4.000 miljoen US-dollar zal opbrengen (zie 'Figuur 3') (Cramer-Flood & eMarketer, 2020).



Figuur 3: Wereldwijde opbrengst door e-commerce verkopen in miljoenen US-dollars (Cramer-Flood & eMarketer, 2020).

Volgens Comeos zijn de voornaamste redenen om online aan te kopen: de prijs, het gemak, het brede aanbod en de snelle leveringsperiode. Zo gaf in het onderzoek van Comeos de helft van de ondervraagden aan online te kopen omdat het gemakkelijker is. Vervolgens geven respectievelijk 42%, 30% en 25% van de ondervraagden aan dat een betere prijs, een breder assortiment en een snelle levering hen motiveert om online te kopen (Comeos, 2020). Het is relevant om deze verwachtingen in te vullen, want het gevoel van voldoening na een aankoop is een motivator om opnieuw bij eenzelfde bedrijf aan te kopen (Kim et al., 2009).

Naast bovenstaande verwachtingen is door het gemak van e-commerce een trend ontstaan om steeds frequenter, maar in kleinere hoeveelheden te bestellen (Gong & De Koster, 2008). Deze stijging in het aantal bestellingen legt extra druk op het leveringssysteem waarbij zo snel mogelijk leveren een vereiste is. Daarenboven kan het leveringsproces nog verder gepersonaliseerd worden door de klant bepaalde voorkeuren zoals tijdstippen of dagen te laten kiezen (Holland, 2019). Het optimaliseren van het leveringsproces wordt bijgevolg zeer complex en moet eveneens kostenefficiënt gebeuren, want klanten willen liefst een goedkope verzendprijs of zelfs een gratis levering. Het belang van snelle, gratis leveringen wordt aangetoond in een Amerikaans onderzoek uit 2019. In dat onderzoek geeft namelijk 44% van 1.188 ondervraagden aan al ooit een bestelling niet te hebben voltooid doordat de leveringstermijn niet binnen de gewenste periode was en zelfs 99% heeft al eens een bestelling niet voltooid omdat de verzending niet gratis was. Tevens bij het terugsturen/retourneren van bestelde producten werd als belangrijkste factor aangegeven dat de retournering gratis is (Tabitha, 2019).

Nadat een klant een online bestelling heeft geplaatst, moeten de volgende processen nog worden doorlopen: betalingsverwerking, orderafhandeling, productlevering en productretourafhandeling (Bayles & Bhatia, 2000). Orderafhandeling en productretourafhandeling behoren tot het takenpakket van een magazijn. Dat is een faciliteit waar activiteiten worden uitgevoerd zoals: ontvangen van goederen, stockage van goederen, orderverzameling en verzending van orders. Een magazijn werkt bijgevolg nauw samen met een leveringsmaatschappij die de productlevering verzorgt (Gu et al., 2007). Merk op dat de producten gestockeerd in een magazijn vaak stock keeping units of SKUs worden genoemd. Een SKU is een product of een bepaalde maat of model van een product dat een bedrijf verkoopt (Cambridge: Cambridge University Press., 1995).

Zoals eerder besproken, staan bedrijven voor een uitdaging om snelle, kleine leveringen voor een aantrekkelijke prijs aan te bieden en zullen ze daarom operationele kosten moeten minimaliseren en inefficiënties oplossen of vermijden. Echter, zoals aangegeven in het onderzoek van Comeos (2020), blijft het aanbieden van een breed gamma belangrijk. Een reorganisatie van de magazijn lay-out en/of magazijntaken kan een oplossing bieden om operationele kosten te verminderen. Een voorbeeld van een reorganisatie van magazijn lay-out is de trend om magazijnrekken steeds hoger te bouwen zodat meer kan worden gestockeerd op eenzelfde grondoppervlak. Dat is vooral een relevante toepassing in steden, waarbij de beschikbare grondoppervlaktes klein en duur zijn en een magazijn kort bij de stadskern de levertijden sterk reduceert (Holland, 2019; Van Dooren, 2020). Een voorbeeld van een reorganisatie van magazijntaken is het optimaliseren van order picking proces. Order picking is het proces waarbij goederen uit een magazijn worden verzameld op aanvraag/bestelling van een klant (K. Roodbergen & Vis, 2006). Het proces is arbeidsintensief of kapitaalintensief (door dure machines) en is daarom gewoonlijk de grootste operationele kost in een magazijn (Gu et al., 2007; Tompkins et al., 2010). Innovatieve toepassingen worden bijgevolg steeds gezocht om deze kost te verminderen.

Scattered storage (of mixed-shelves storage) is een voorbeeld van een innovatieve toepassing om het order picking proces te optimaliseren. Deze strategie wordt toegepast binnen magazijnen van B2C e-commerce bedrijven. Producten worden daarbij als afzonderlijke artikels over de reklocaties verspreid en bevinden zich daardoor op meerdere plaatsen in het magazijn (Weidinger, 2018). Dat is in tegenstelling tot traditionele opslagmethoden, zoals dedicated storage, waarbij elk product slechts in een beperkt gebied van het magazijn beschikbaar is en eventueel zelfs een vaste locatie toegewezen krijgt (Weidinger & Boysen, 2018). Door scattered storage toe te passen, worden kleine orders significant sneller gepickt en ontvangt de klant bijgevolg sneller de gemaakte bestelling. Een nadeel van deze strategie is dat het order picking probleem toeneemt in complexiteit. Naast het bepalen van de route doorheen het magazijn, moeten nu eveneens de locaties om producten te picken worden bepaald. Daarnaast zullen reklocaties snel leeg zijn, doordat telkens maar één artikel of een beperkt aantal op een locatie beschikbaar is. Deze lege reklocaties moeten regelmatig opnieuw bijgevuld worden (Weidinger et al., 2018).

1.2 Probleemstelling

Nog maar weinig onderzoek werd uitgevoerd naar scattered storage. Dat zorgt voor een breed gamma van onderzoeksmogelijkheden. Door Weidinger werd in 2018 onderzocht hoe het pickproces met slechts één picker best georganiseerd kan worden. Meer specifiek werden drie heuristieken ontwikkeld die het picker-routing probleem oplossen. Dat probleem bestaat uit twee stappen. De eerste stap is het selecteren van de te bezoeken opslaglocaties, want SKUs zijn op meerdere locaties in het magazijn beschikbaar. De tweede stap vormt een route tussen deze locaties (Weidinger, 2018). Het doel van deze masterproef is de heuristieken verder te onderzoeken. Ze zullen worden getest met een nieuwe dataset en eventuele verbeteringen worden onderzocht. Daardoor worden de onderzoeksvraag en bijbehorende deelvragen bekomen:

Onderzoeksvraag: 'Hoe kan in een scattered storage picker-to-parts magazijn van een B2C e-commerce bedrijf het pickproces optimaal georganiseerd worden?'

Deelvraag 1: 'Welke factoren beïnvloeden het pickproces?'

Deelvraag 2: 'Wat is het bestaande onderzoek naar scattered storage?'

Deelvraag 3: 'Welke toepassingen zijn gelijkaardig aan het scattered storage picker-routing probleem?'

Deelvraag 4: 'Wordt bij het testen van de heuristieke oplossingsmethoden voor het picker-routing probleem op nieuwe data dezelfde oplossingen gevonden als in de literatuur?'

Deelvraag 5: 'Hoe kan bepaald worden welke orders een picker tegelijkertijd zal picken?'

Deelvraag 6: 'Hoe kan de selectie van opslaglocaties voor het picker-routing probleem opgelost worden?'

De onderzoeksvraag wordt verder afgebakend door enkel te focussen op bedrijven die actief zijn in de B2C en e-commerce sector en een manueel picker-to-parts verzamelsysteem toepassen. Dat is een verzamelsysteem waarbij de pickers zich doorheen het magazijn verplaatsen naar de opslaglocaties van de te picken SKUs. Deze afbakening werd gekozen doordat magazijnen vaak zo een manueel order picking systeem toepassen (C. G. Petersen, 2002). Volgens de Koster et al. (2007) is dat zelfs 80% van de West-Europese magazijnen.

De onderzoeksvraag wordt ondersteund door zes deelvragen. Ten eerste wordt onderzocht welke factoren het pickproces met meerdere pickers beïnvloeden. Ten tweede wordt het huidige onderzoek over scattered storage onder de loep genomen. Ten derde wordt gezocht naar toepassingen die gelijkaardig zijn aan het picker-routing probleem in een scattered storage magazijn. Ten vierde worden heuristieke oplossingsmethoden voor het picker-routing probleem afkomstig uit de literatuur getest op een nieuwe dataset om na te gaan of de gevonden resultaten overeenkomen met die uit de literatuur. Ten vijfde wordt getest of batching voordelig kan zijn om toe te passen op picklijsten in een scattered storage magazijn en welke batching methoden dan best kunnen toegepast worden. Ten slotte wordt nagegaan of naast de huidige prioriteitsheuristieken nog een andere methode gedefinieerd kan worden om het opslaglocatie-selectie probleem efficiënt op te lossen.

1.3 Methodologie

In dit deel wordt concreet besproken hoe de onderzoeksvragen zullen worden beantwoord. Eerst wordt de nodige kennis over het thema onderzocht door een literatuurstudie. Vervolgens zal in de praktijkstudie deze kennis gebruikt worden om mathematische formuleringen op te stellen en op te lossen.

1.3.1 Literatuurstudie

In de literatuurstudie wordt een grote hoeveelheid literatuur over zowel tactische beslissingen als operationele beslissingen in magazijnen verzameld en geanalyseerd. Literatuur kan onder meer papers en boeken zijn. Het doel hiervan is om bekend te worden met de theoretische achtergrond van magazijnen en zo de onderzoeksvragen onderbouwd te kunnen beantwoorden. De literatuurstudie zal bestaan uit drie onderdelen. Eerst wordt een beeld geschetst van alle beslissingen binnen een magazijn. Dat houdt in: de lay-out van het magazijn, routeringsbeslissingen, order batching beslissingen en de indeling van producten in de rekken. Als tweede deel zal het huidig onderzoek over scattered storage bestudeerd worden. Ten slotte, als derde deel, wordt gezocht naar gelijkaardige toepassingen in andere domeinen die hulp kunnen bieden bij het oplossen van de onderzoeksvragen, zoals bijvoorbeeld transportvraagstukken.

1.3.1.1 Zoektermen

Voordat literatuur kan verzameld worden, moeten de zoektermen worden bepaald. Hierdoor kan met een gerichte aanpak gezocht worden naar literatuur over het afgebakende thema. Bij het opstellen van de lijst met zoektermen wordt er rekening gehouden met synoniemen door gebruik te maken van de website Thesaurus[6]. De lijst met zoektermen zal voornamelijk bestaan uit Engelse termen, maar vertalingen naar het Nederlands zijn eveneens mogelijk. Ten slotte worden de gevonden zoektermen gecombineerd tot verschillende mogelijke zoekopdrachten, zoals:

- (Scattered storage OR mixed shelve storage) AND (e-commerce OR ecommerce OR electronic commerce)
- (Scattered storage OR mixed shelve storage) AND (e-commerce OR ecommerce OR electronic commerce) AND (picking process OR picking)
- (Scattered storage OR mixed shelve storage) AND (e-commerce OR ecommerce OR electronic commerce) AND (B2C OR business-to-consumer)
- (Gemengde opslag OR willekeurige opslag) AND (e-commerce OR ecommerce)
- …

1.3.1.2 Databases

Als databases om de zoekopdrachten uit te voeren wordt gebruikt gemaakt van: Google Scholar, EBSCOhost en Bibliotheek UHasselt.

1.3.1.3 Inclusie- en exclusiecriteria

Inclusiecriteria zijn eigenschappen waaraan de gevonden literatuur moet voldoen om opgenomen te worden in de literatuurlijst. Toegepast op het afgebakende thema zijn dit de volgende criteria:

- De literatuur komt van een betrouwbare bron. Een duidelijke vermelding van de oorsprong van de gegevens van een betrouwbare instelling is beschikbaar of het is geschreven door een autoritaire auteur binnen het vakgebied.
- De gevonden informatiebronnen hebben liefst eveneens een peer-reviewproces doorlopen. Maar dat is geen vereiste.

Exclusiecriteria zijn eigenschappen waaraan de gevonden literatuur voldoet om niet opgenomen te worden in de literatuurlijst. Toegepast op het afgebakende thema zijn dit de volgende criteria:

- Literatuur die niet is geschreven in het Nederlands of in het Engels.
- Literatuur die zich dateert van voor 1995. (Zie '1.3.1.4 Afbakening van de tijdsperiode' voor verder uitleg.

[6] Thesaurus is een website waarmee synoniemen kunnen gevonden worden [www.Thesaurus.com].

1.3.1.4 Afbakening van de tijdsperiode

Zoals beschreven in de exclusiecriteria (zie '1.3.1.3 Inclusie- en exclusiecriteria') wordt niet gekeken naar literatuur uitgebracht voor het jaartal 1995. Dat is bepaald door rekening te houden met de geschiedenis van e-commerce. In 1995 werd namelijk Amazon.com opgericht dat destijds via het internet boeken aan consumenten verkocht. Ze waren het eerste bedrijf waarbij kleine orders (boeken) moesten gepickt worden uit een magazijn (Mellahi & Johnson, 2000).

Indien de inhoud van bepaalde literatuur geen last heeft van veroudering, zoals ongewijzigde theorie, dan wordt toch gebruik gemaakt van bronnen voor 1995. Eveneens geldt dit indien opzettelijk wordt verwezen naar gebeurtenissen in de geschiedenis daterend voor 1995.

1.3.1.5 Identificatie van bijkomende literatuur

Naast literatuur gevonden via de lijst van zoektermen en databases, wordt gebruikt gemaakt van 'citation chaining' om bijkomende literatuur te verzamelen. Hierbij wordt zowel gekeken naar papers geciteerd door de originele paper als naar papers die de originele paper hebben geciteerd.

1.3.2 Praktijkstudie

Het resultatengedeelte kan op twee verschillende manieren worden uitgewerkt. Na het uitwerken van de literatuurstudie zal nog gekozen worden tussen ofwel een theoretische uitwerking of een praktische uitwerking bij een bedrijf.

1.3.2.1 Theoretische uitwerking

Een theoretische uitwerking van de onderzoeksvragen houdt in dat de wiskundige modellen zullen worden opgelost met artificiële kwantitatieve data. In dat geval kan eveneens een simulatie worden opgesteld over een denkbeeldig magazijn om een visueel voorbeeld op te stellen.

Sommige bedrijven, zoals Amazon dat scattered storage toepast, bieden rondleidingen in hun magazijn aan. Dat is een mogelijke methode om informatie uit de praktijkwereld te verzamelen en zo de artificiële data realistisch op te stellen.

1.3.2.2 Praktische uitwerking

Bij een praktische uitwerking van de onderzoeksvragen zal worden samengewerkt met een bedrijf. In dat geval doen er zich twee mogelijkheden voor: het bedrijf past al scattered storage toe of het bedrijf past dat nog niet toe. Indien het bedrijf al scattered storage toepast, zal worden onderzocht welke verbeteringen nog kunnen worden toegepast via wiskundige modellen en een simulatiemodel op basis van kwantitatieve data van het bedrijf. Indien het bedrijf nog geen scattered storage toepast, dan wordt een denkbeeldige uitwerking van scattered storage bedacht op basis van de kwantitatieve data van het bedrijf.

Bedrijven zullen worden gecontacteerd via telefonisch contact of via mail, en dat zal altijd gebeuren met goedkeuring van de promotor Kris Braekers. Het bedrijf moet dan interesse tonen in het project en het moet voldoen aan de volgende criteria:

- Het bedrijf beschikt over minstens één magazijn.
- Het bedrijf is actief in de B2C sector.
- Het bedrijf past scattered storage al toe of heeft interesse om dat in de toekomst toe te passen.

Eens een samenwerking tot stand komt, dienen een aantal afspraken gemaakt te worden door de volgende vragen te stellen:

- Zijn interviews mogelijk met werknemers betrokken bij het pickingproces of bij werknemers die verantwoordelijk zijn voor de operationele werking van het magazijn?
- Bestaat de mogelijkheid om het magazijn van het bedrijf te bezoeken?
    - Bestaat de mogelijkheid tot een tweede bezoek indien nodig?
- Wie is de contactpersoon voor eventuele vragen? Of zijn dat meerdere contactpersonen?
- …

Mogelijke bedrijven die kunnen gecontacteerd worden zijn: Amazon.com, Zalando, Bol.com, Nike, Adidas, ZEB, Mediamarkt, Coolblue, enzoverder.

1.3.2.3 Inhoud kwantitatieve data

De kwantitatieve data moet over volgende elementen beschikken:

- De afmetingen van het magazijn.
- De afmetingen van de rekken.
- De locaties van de rekken.
- De breedte van de gangpaden (oftewel de ruimte tussen de rekken).
- Het aantal SKUs en afmetingen van de SKUs.
- De ligging van het depot of van meerdere depots.
- Het aantal beschikbare order pickers.
- Het werkschema van de order pickers.
- Het aantal bestellingen per tijdseenheid en het gemiddelde aantal SKUs in een bestelling.

Deze data wordt ofwel bij een bedrijf verzameld of zal fictief worden opgesteld.

2 Literatuurstudie

Om de onderzoeksvragen onderbouwd en volledig te kunnen beantwoorden, wordt in de literatuurstudie een beeld geschetst van de theoretische achtergrond van magazijnen. Concreet zullen drie delen worden besproken. Eerst worden de basisbeslissingen in een magazijn onderzocht, zoals stockering, routering, etc. Vervolgens wordt in meer detail het huidige onderzoek over scattered storage besproken. Ten slotte worden gelijkaardige toepassingen in andere domeinen besproken, zoals bijvoorbeeld transportvraagstukken, die hulp kunnen bieden bij het oplossen van de onderzoeksvragen.

2.1 Beslissingen in een magazijn

Zowel op strategisch als operationeel vlak moeten in een magazijn tientallen beslissingen genomen worden. Dat kan gaan van de constructie van het magazijn tot het beslissen waar een SKU gestockeerd zal worden. Desbetreffende beslissingen zullen besproken worden in twee grote onderdelen, namelijk de lay-out van het magazijn en de operationele werking van het magazijn (Gu et al., 2007).

2.1.1 Lay-out van het magazijn

De lay-out van het magazijn wordt bepaald door strategische beslissingen. Dat betekent dat deze beslissingen over een lange termijn vastliggen en een grote invloed hebben op de werking van het magazijn (Gu et al., 2010). Geïnspireerd op de paper van Gu et al., (2007) wordt de lay-out van het magazijn opgesplitst in vier onderdelen: de structuur, de afmetingen, de departementen en ten slotte de apparatuur.

2.1.1.1 Structuur van de magazijnhal

Om de structuur of de vormgeving van de hal te bepalen, rekening houdend met de opslag- en doorvoervereisten, moeten beslissingen genomen worden zoals: de locatie van de nooduitgang(en), de locatie van de laad- en losdeuren, de vorm van de magazijnhal (bijv. rechthoekig), enzoverder (Gu et al., 2010). Daarnaast heeft de structuur van de magazijnhal een invloed op beslissingen die later zullen worden toegelicht zoals: het aantal opslagafdelingen, de gebruikte apparatuur en de order picking methode (Gu et al., 2010).

Een voorbeeld van een beslissing binnen de structuurbepaling van de hal is de plaatsing van laad- en losdeuren. Via een laaddeur kunnen uitgaande zendingen geladen worden in een transportmiddel en via een losdeur kunnen binnenkomende zendingen gelost worden in de magazijnhal. Concreet moet bepaald worden waar deze deuren zich zullen bevinden, het aantal deuren en de positie van de deuren. Het aantal laad- en losdeuren is afhankelijk van de laad- en losduur en het aantal vrachtwagens per tijdsbestek. De positie van elke deur kan recht of schuin zijn. De rechte positie heeft als voordeel dat een vrachtwagen zowel vanaf de linkerkant als vanaf de rechterkant kan komen aanrijden. De schuine positie heeft als voordeel dat een vrachtwagen op de parking minder manoevreerruimte zal nodig hebben om zich tegen de loskade te plaatsen (Engelbregt & Kruijer, 2009).

2.1.1.2 Afmetingen van de magazijnhal

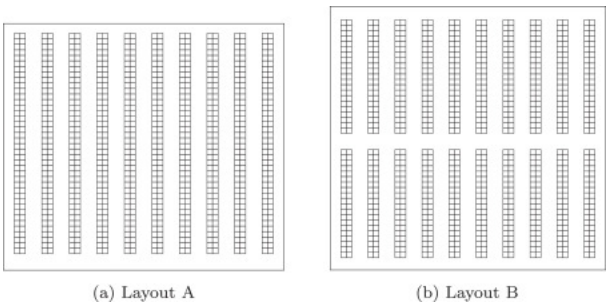
De afmetingen (m.a.w. de grootte) van de magazijnhal bepalen de capaciteit van een magazijn. Om deze afmetingen te bepalen, moeten een aantal kostenaspecten in rekening worden gebracht: de constructiekosten, de opslagkosten en de kosten verbonden aan het niet behalen van de opslagvraag. Deze kosten zijn van toepassing voor zowel externe als interne magazijnen. Indien een bedrijf beschikt over een intern magazijn, d.w.z. geen uitbesteding van de magazijntaken, dan heeft het bedrijf logischerwijs invloed op het voorraadbeleid. Het interne magazijn is dan op de hoogte van de inkomende SKUs en kan de voorraadsniveau's onder controle houden door te communiceren met de andere departementen binnen het bedrijf. Een intern magazijn moet daarom nog bijkomende kosten in rekening brengen, namelijk kosten verbonden aan het voorraadbeleid (Gu et al., 2007).

De breedte en de lengte dimensie bepalen de opppervlakte van een magazijn, maar de hoogte is eveens een belangrijke factor. De hoogte van het gebouw bepaalt namelijk hoe hoog de stockage rekken kunnen zijn en welk type vervoersmiddel (zie '2.1.1.4 Selectie van de apparatuur') gebruikt zal worden om de SKUs in de rekken te leggen of uit de rekken te halen (Engelbregt & Kruijer, 2009). Daarnaast heeft de hoogte van het gebouw een invloed op de keuze van de vloer. Hoe hoger het gebouw, hoe groter de draagkracht van de vloer moet zijn. Bijvoorbeeld betonvloeren hebben een zeer grote draagkracht (De Kelver, 2006).

2.1.1.3 Lay-out van de departementen

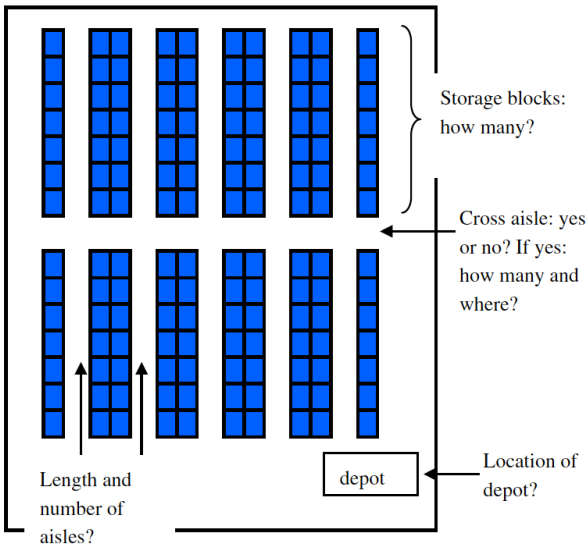
De lay-out van de departementen betreft de beslissing waar de verschillende afdelingen worden gevestigd (bijv. ontvangst, opslag, sortering, verzending, enzoverder). Deze keuze houdt rekening met de relaties tussen de departementen en tracht de material handling kosten[7] te minimaliseren (de Koster et al., 2007). Volgens de afbakening van de masterproef is het interessant om de opslagafdeling in meer detail te bespreken.

Om de lay-out van de opslagafdeling te bepalen, moeten verschillende keuzes gemaakt worden over blokken, rekken, gangpaden en/of depot(s), zoals aangeduid op 'Figuur 5'. Elk van deze keuzes wordt in één van de onderstaande alinea's kort besproken.



Figuur 4: Voorbeelden van een magazijn lay-out met en zonder kruisgang (Öztürkoğlu & Hoser, 2019).

[7] De material handling kosten zijn vaak een lineaire functie van de afgelegde afstand met SKUs doorheen het magazijn (de Koster et al., 2007).



Figuur 5: Voorbeeld van een lay-out van de opslagafdeling en de bijbehorende beslissingen (de Koster et al., 2007).

Een blok is een verzameling van opslagrekken (de Koster et al., 2007) en in een picker-to-parts magazijn bevinden deze rekken zich typisch parallel t.o.v. elkaar (Scholz et al., 2016). Tussen aanliggende blokken bevindt zich een kruisgang, zoals in 'Figuur 5', en daardoor kan een order picker makkelijker van pickgang veranderen, zoals in lay-out B in 'Figuur 4'. Een pickgang is een gangpad tussen twee rekken (Scholz et al., 2016). Indien geen blokken worden gevormd in het magazijn, dan wordt de ruimte beter benut. De opslagrekken worden dan langer(Gu et al., 2010), zoals in lay-out A in 'Figuur 4'. Een keuze tussen een hogere toegankelijkheid of een betere benutting van de ruimte kan volgens Guo et al. (2016) afhankelijk zijn van de vraag naar SKUs. Zo heeft bijvoorbeeld een brede en ondiepe lay-out (gekenmerkt door veel korte gangen) de voorkeur als de SKUs over een vergelijkbare vraag beschikken. Daarentegen heeft een smalle en diepe lay-out (gekenmerkt door weinig lange gangen) de voorkeur als de SKUs over een verschillende vraag beschikken (Guo et al., 2016).

Rekken worden gebruikt om SKUs in te stockeren en kunnen verschillen in afmetingen (lengte, breedte en hoogte). In Europa zijn bijvoorbeeld in 80% van de magazijnen de opslagrekken tot op hoofdhoogte en zijn daardoor toegangkelijk voor manuele order picking zonder lift (de Koster et al., 2007; Weidinger, 2018). De hoogte van een rek en het aantal leggers bepalen samen het aantal verdiepingen in datzelfde rek, en de hoogte tussen de leggers en de afstand tussen de SKUs geeft de grootte van een stocklocatie weer (Wolf et al., 2018).

Gangpaden worden gebruikt om van pickgang te wisselen. Ze bevatten geen opslaglocaties (Scholz et al., 2016). Voorbeelden van beslissingen over de gangpaden zijn: de oriëntatie van de gangpaden, het aantal gangpaden, de lengte en breedte van de gangpaden en ten slotte de ingang(en) (Gu et al., 2010). De breedte van een gangpad heeft invloed op de verplaatsingsmethode van order pickers. Indien een gang voldoende breed is, dan kunnen order pickers namelijk elkaar kruisen en is bijgevolg tweerichtingsverkeer mogelijk. Bovendien bepaalt de breedte van de pickgang eveneens of een picker moet uitstappen om op dezelfde positie langs beide zijden van de gang een SKU te kunnen picken (K. J. Roodbergen & De Koster, 2001).

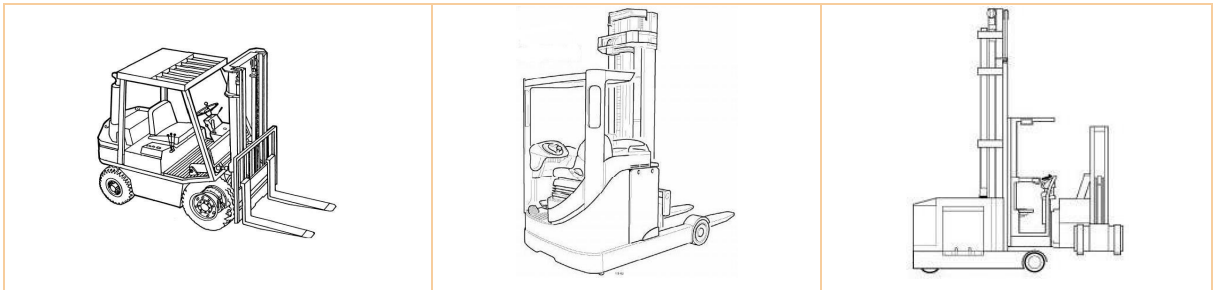
Het depot is het start- en/of eindpunt van de picking routes (K. Roodbergen & Vis, 2006) en kan zich zowel centraal als decentraal in het magazijn bevinden (Theys et al., 2010). Het aantal depots en de locatie van de depot(s) moeten eveneens worden bepaald. De toegankelijkheid van de depot(s) heeft namelijk een invloed op de route die een order picker moet maken. Indien een depot altijd dichtbij is, dan moet de order picker geen onnodige langere afstand afleggen (Weidinger & Boysen, 2018).

In het opslagdepartement van een magazijn kunnen de SKUs verder worden opgedeeld in afdelingen met elk een eigen lay-out. Redenen om dat te doen zijn: de fysieke eigenschappen van SKUs (bijv. palletvorm), management overwegingen (bijv. per klant) of material handling overwegingen (bijv. buffervoorraad) (Gu et al., 2007). Een voorbeeld van twee aanvullende afdelingen binnen het opslagdepartement zijn de buffervoorraad en de grijpvoorraad. De buffervoorraad is een opslaggebied waar SKUs nog niet uit de bulkverpakking zijn gehaald en bijgevolg in grotere eenheden gestockeerd worden. De nadruk in de buffervoorraad ligt op ruimtebenutting. De buffervoorraad wordt vervolgens gebruikt om de grijpvoorraad aan te vullen waarbij de SKUs direct bereikbaar zijn (uit de verpakking) voor de order pickers (Engelbregt & Kruijer, 2009).

2.1.1.4 Selectie van de apparatuur

De selectie van de apparatuur in een magazijn bepaalt de mate van automatisatie, de mogelijke stockeringsmethoden en de verplaatsingsmethodes (Gu et al., 2010). Alle aparatuur in een magazijn vormt tesamen een lange lijst, maar het valt buiten de afbakening van de masterproef om deze allemaal te bespreken. Enkel de apparatuur gebruikt tijdens het picker-to-parts routeringsproces zal daarom worden besproken.

De selectie van de apparatuur in het picker-to-parts routeringsproces is afhankelijk van de lay-out van het magazijn. In dit deel zullen een aantal voorbeelden worden besproken a.d.h.v. hoogte van het magazijngebouw. In een traditioneel magazijn (< 8 meter hoog) wordt meestal gebruik gemaakt van een normale heftruck. Indien de gangpaden smaller zijn dan 3,20 meter dan is een reachtruck nodig. In een middelhoogbouwmagazijn (8-12 meter hoog) of een hoogbouwmagazijn (> 12 meter hoog) kan gebruik gemaakt worden van een turrettruck. Deze voertuigen bewegen zich railgebonden voort tussen de rekken (d.w.z. ze kunnen niet afwijken van een magnetisch spoor in de magazijnvloer) (Engelbregt & Kruijer, 2009).



Tabel 1: Voorbeelden van respectievelijk een heftruck, een reachtruck en een turrettruck (Belgreen, z.d.; Lesboek heftruck/reachtruck, 2016).

2.1.2 De operationele werking van het magazijn

Binnenkomende zendingen worden naar het magazijn gebracht, bij de dokken wordt het vervoersmiddel gelost en de SKUs worden gestockeerd in de rekken. Ondertussen bestellen klanten SKUs. De bestelde SKUs worden uit de rekken gehaald. Het pakket wordt voorbereid en via de dokken geladen in een vervoersmiddel om naar de klant te verzenden. In het bovenstaande proces zijn volgens Gu et al. (2007) drie beslissingsgebieden te herkennen: ontvangen & verzenden, opslag en order picking. Deze beslissingsgebieden kunnen worden beïnvloed door het type order. Een order kan verschillen op vier vlakken: prioriteit (bijv. spoedorders), samenstelling, verpakkingswijze en verzendwijze (Engelbregt & Kruijer, 2009).

2.1.2.1 Ontvangen en verzenden

Ontvangst- en verzendactiviteiten omvatten onder andere de toewijzing van vrachtwagens aan dokken, het plannen van laad- en losactiviteiten en kwaliteitsinspectie. Deze activiteiten zijn afhankelijk van zowel het stockeringsproces als het order pickingsproces. Zo kan een vrachtwagen bijvoorbeeld pas geladen worden indien de noodzakelijke SKUs gepickt zijn (de Koster et al., 2007; Gu et al., 2007).

2.1.2.2 Stockering

De opslagafdeling bevat alle gestockeerde SKUs. Deze afdeling kan worden opgedeeld in departementen en vervolgens opnieuw in zones. In deze departementen/zones liggen de SKUs in een opslaglocatie (Gu et al., 2007).

A SKU departementen

Het opdelen van de opslagafdeling in departementen werd al besproken in '2.1.1.3 Lay-out van de departementen'. Na het opstellen van de verschillende departementen moet nog beslist worden hoe de departementen zullen gevuld worden met SKUs (Gu et al., 2007). Meer specifiek zal bepaald moeten worden welke SKU in welk departement gestockeerd wordt en in welke hoeveelheid. Bijvoorbeeld een departement wordt gevuld met alle SKUs die aan een specifieke klant toebehoren. Het is eveneens mogelijk dat een SKU zich in meerdere departementen bevindt (Gu et al., 2007).

B Zoning

Een pick zone is een verzameling van opslaglocaties die in elkaars buurt liggen (Gu et al., 2007). De grootte van de pick zones is afhankelijk van het aantal SKUs in het magazijn, het aantal order pickers en de beschikbare tijd voor orderverwerking (C. G. Petersen, 2002). Order pickers worden toegewezen aan één of meerdere zones. Daardoor worden ze bekend met de opslaglocaties van de SKUs in hun toegewezen zone en moeten ze slechts een kleine afstand afleggen om SKUs te picken. Het kan nadelig zijn dat een picker in zijn/haar zone moet blijven, want hierdoor moeten orders soms parallel of sequentieel gepickt worden en dat zorgt voor extra kosten (Gu et al., 2007; Jane & Laih, 2005; C. G. Petersen, 2002). Van Gils et al. (2018) onderscheiden twee methodes om SKUs in zones op te delen:

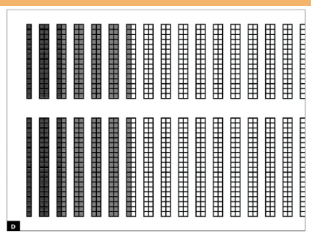
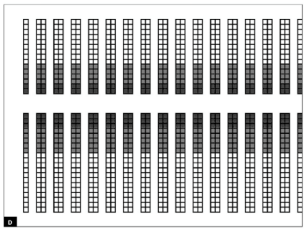
Zoning methode	Uitleg van de methode
Product properties assignment	Toewijzing van SKUs aan zones o.b.v. producteigenschappen (bijv. grootte, gewicht, enz.).
Demand properties assignment	Toewijzing van SKUs aan zones o.b.v. vraageigenschappen (bijv. klanttype, orderfrequentie, enz.).

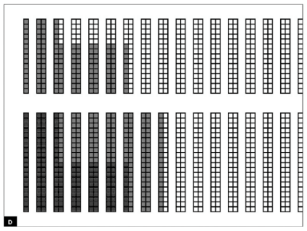
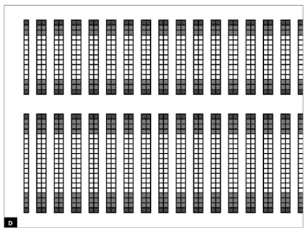
Tabel 2: Overzicht met voorbeelden van zoning methoden (de Koster et al., 2007; C. G. Petersen, 2002; van Gils et al., 2018).

C Locatiebeslissingen van SKUs

Nadat de opslagafdeling werd opgedeeld in departementen en mogelijks opnieuw in zones, moet beslist worden op welke exacte opslaglocatie een SKU terecht zal komen. Een SKU kan dan een vaste locatie (dedicated storage) krijgen toegewezen of een verzameling van mogelijke vrije locaties (de Koster et al., 2007; Engelbregt & Kruijer, 2009).

Een voorbeeld van een opslagstrategie is class-based storage, waarbij de SKUs worden opgedeeld in klasses volgens een parameter, zoals vraag, omzet of verblijfsduur (Cardona & Gue, 2020; Guo et al., 2016). Er wordt bijvoorbeeld gewerkt met drie omzetklasses: klasse A, klasse B en klasse C. Hierbij bevat klasse A de SKUs met een hoge omzet, klasse B bevat de SKUs met een matige omzet en klasse C bevat de SKUs met een lage omzet (van Gils et al., 2018). De vastgelegde klasses kunnen vervolgens via een gekozen methode worden opgeslagen in de rekken, zoals de voorbeelden in 'Tabel 3: '. Een rode draad bij deze methodes is dat SKUs uit klasse A toegankelijker of dichter bij het depot zullen worden gestockeerd dan SKUs uit klasse C. Binnen de toegewezen opslaglocaties aan een klasse, worden de SKUs uit die klasse willekeurig verdeeld (Yu et al., 2015). Het nadeel van deze methode is dat de vraag naar SKUs constant varieert en de klasses daarom periodiek herzien zouden moeten worden (de Koster et al., 2007).

Methode van class-based storage	Uitleg van de methode	Visueel voorbeeld (Klasse A = donkergrijs; klasse B = lichtgrijs; klasse C = wit & depot = D)
Within-aisle	Elke pickgang bevat SKUs van eenzelfde klasse. Bijgevolg wordt het pickverkeer geconcentreerd in gangen kortst bij het depot.	
Across-aisle	Elke klasse is gelegen over verschillende pickgangen.	

Diagonal	De klassen zijn gelegen in verhouding tot de afstand tot het depot. Bijgevolg wordt het pickverkeer geconcentreerd in gangen kortst bij het depot.	
Perimeter	De belangrijkste klassen bevinden zich aan de uiteindes van de opslagrekken.	

Tabel 3: Overzicht met voorbeelden van opslagmethoden binnen een class-based storage strategie (de Koster et al., 2007; van Gils et al., 2018).

Indien de SKUs worden opgedeeld in slechts één klasse, dan past men random storage toe (Guo et al., 2016). Bij de toepassing van deze strategie wordt de opslaglocatie van de SKU willekeurig geselecteerd uit alle beschikbare lege opslaglocaties. Random storage is eenvoudig toe te passen en daarom vaak gebruikt. Daarnaast zorgt deze methode voor een gebalanceerde vulling van het magazijn (C. G. Petersen, 2002), maar eveneens voor hogere material handling kosten (de Koster et al., 2007).

Naast de besproken opslagstrategieën bestaan nog veel meer mogelijkheden zoals toewijzen op basis van cube-per-order index (COI), naar volume, naar gewicht, per familiegroep, per producteigenschappen, enzoverder (Engelbregt & Kruijer, 2009; Weidinger, 2018). Eveneens scattered storage, het onderwerp van deze masterproef, is een opslagmethode die later in de literatuurstudie nog concreet zal worden besproken.

De locatiebeslissingen hebben als doel de ruimte in de rekken maximaal te benutten en de material handling te minimaliseren (Gu et al., 2007). De beslissingen hebben eveneens een significante impact op het order picking proces (Gu et al., 2007). Indien goederen uit eenzelfde bestelling dichter bij elkaar liggen, dan zal deze bestelling sneller kunnen worden samengesteld (Weidinger & Boysen, 2018). Om gebruik te maken van dat voordeel kan de family-grouping strategie worden toegepast waarbij vaak samen bestelde SKUs dichter bij elkaar gestockeerd worden (de Koster et al., 2007).

2.1.2.3 Order picking

Klanten bestellen kleine volumes van SKUs en terwijl komen deze SKUs in grote hoeveelheden aan in magazijnen. Om deze hoeveelheidsverschillen te overbruggen bestaat de functie order picking (van Gils et al., 2018). Tijdens het order picking proces worden de door klanten bestelde SKUs verzameld uit de voorraad van het magazijn. Een klantenorder bestaat hierbij uit één of meerdere orderlijnen, waarbij elke orderlijn een aantal items van een SKU vertegenwoordigt (de Koster et al., 2007).

Het order picking proces gebeurt meestal ordergeoriënteerd d.w.z. picken per order (oftewel discrete picking) (de Koster et al., 2007). Hierbij kan een order door één of meerdere pickers worden samengesteld en een picker verzamelt één order of meerdere orders tegelijkertijd. In andere gevallen vindt het order picking proces artikelgeoriënteerd plaats. Hierbij worden de orders ontleedt tot artikels en worden artikels die kort bij elkaar liggen in het magazijn samen gepickt. Nadien moeten de artikels via een sorteersysteem worden samengevoegd tot een order (Engelbregt & Kruijer, 2009). Indien een order picker de SKUs in zijn/haar picking kar sorteert tijdens het pickproces, dan wordt de sort-while-pick methode toegepast. Indien de SKUs na het pickproces moeten samengesteld worden tot één order, dan wordt de sort-after-pick methode toegepast. Om deze laatste methode toe te passen is een sorteersysteem noodzakelijk (Gu et al., 2007). Verschillende types sorteersystemen zullen niet verder worden besproken, omdat dat buiten de afbakening van de masterproef valt.

Om het order picking process te optimaliseren, kunnen vier componenten onderscheiden worden die samen de orderpicktijd bepalen. Deze componenten zijn: de insteltijd (voorbereidingstijd), de verplaatsingstijd, de zoektijd naar een opslaglocatie en de picktijd om de SKU uit het rek te halen. De verplaatsingstijd beïnvloedt het grootste deel van de orderpicktijd. Dat is logisch, want de resterende componenten (insteltijd, zoektijd en picktijd) kunnen als constanten worden beschouwd (Scholz et al., 2016). Toepassingen die trachten deze verplaatsingstijd te minimaliseren zijn wave, batch & zone picking en routering (Gu et al., 2007). Deze zullen hieronder worden besproken.

A Wave, batch & zone picking

De verzameling van de SKUs kan op twee manieren gebeuren: parts-to-picker of picker-to-parts. In een parts-to-picker situatie zullen de SKUs naar de order picker toe komen. Daarentegen, zal in een picker-to-parts situatie de order picker zich naar de SKUs gestockeerd in de opslagrekken verplaatsen (Engelbregt & Kruijer, 2009; Scholz et al., 2016). Zoals aangegeven in '1.2 Probleemstelling' wordt in deze masterproef enkel de picker-to-parts situatie onderzocht. Drie vaak gebruikte methodes voor picker-to-parts order picking zijn: wave picking, batch picking en zone picking.

De eerste methode, wave picking, maakt gebruik van tijdsperiodes. Per wave worden de orders gepickt die binnen eenzelfde tijdsperiode moeten klaar zijn (de Koster et al., 2007).

Indien orders klein zijn, dan kan het handig zijn om meerdere orders tegelijk te picken en vooral als dezelfde SKU nodig is voor meerdere bestellingen (de Koster et al., 2007). De tweede methode, namelijk order batching, is zo een methode waarbij de orders op een picklijst worden gegroepeerd (Gu et al., 2007). Een order picker is verantwoordelijk om een batch te picken en zal daardoor mogelijks meerdere orders tegelijkertijd picken. Een picker moet ongeveer dezelfde route doorlopen om deze orders te picken en zo kan de totale reistijd worden geminimaliseerd. Een nadeel is dat een grotere pick-kar, waarin meerdere orders passen, de beweeglijkheid van de picker vermindert (Weidinger et al., 2018). In 'Tabel 4: ' worden vier methoden besproken om batches samen te stellen.

Batchingsmethode	Uitleg van de methode
Priority rule based	Orders worden volgens hun prioriteit op de picklijst gezet. Bijvoorbeeld first-come-first-served (FCFS), waarbij orders die eerst binnen komen, eerst gepickt zullen worden.
Seed based	De batches worden bepaald in twee stappen. De eerste stap, seed selection, bepaalt een eerste order om toe te voegen aan een batch. Bijvoorbeeld door de order te kiezen met de meeste picklocaties. De tweede stap, order congruency, bepaalt welke nog niet-toegewezen order als volgende aan de huidige batch moet worden toegevoegd. Bijvoorbeeld door rekening te houden met het aantal extra gangpaden dat bezocht moet worden als het order wordt toegevoegd.
Savings based	Batches worden bepaald door orders te combineren die de grootste afstandsbesparing teweeg brengen.
Exact algorithm	Een optimale oplossing voor het batching probleem.

Tabel 4: Overzicht met voorbeelden van batching methoden (de Koster et al., 2007; van Gils et al., 2018).

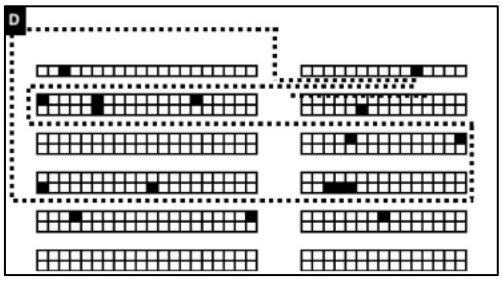
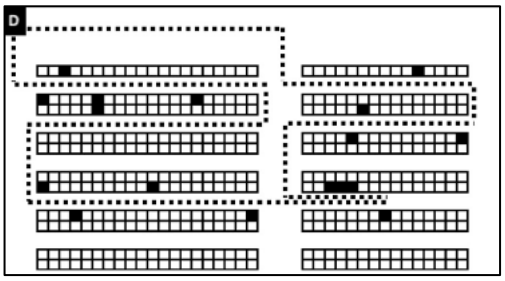
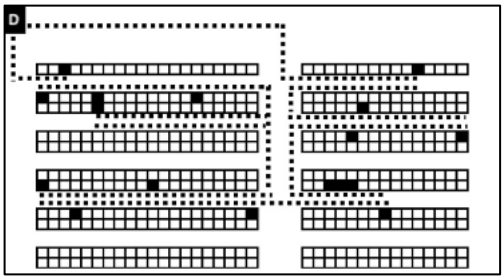
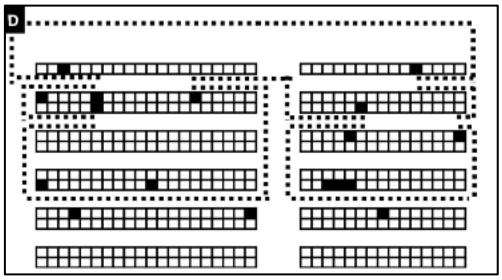
De derde methode, zone picking, maakt gebruik van pick zones. Elke order picker verzamelt steeds SKUs binnen zijn/haar aangewezen zone en brengt de SKUs naar het depot. Indien een order bestaat uit SKUs uit verschillende zones, dan moeten deze nog worden samengesteld tot één pakket. Dat kan gebeuren op twee manieren: sequential zone picking of parallel zone picking. Bij sequential zone picking (of progressive zoning) werkt telkens slechts één order picker aan het samenstellen van het order. De containers met de tot dan gepickte SKUs worden dan steeds doorgegeven naar de volgende zone. Na de laatste zone bevat één container bijgevolg alle SKUs van een order. Nadelig kan zijn dat een order picker moet wachten op de tot dan gepickte SKUs uit een eerdere zone. Bij parallel zone picking (oftewel synchronized zoning) werken meerdere order pickers gelijktijdig aan het samenstellen van het order door gelijktijdig de benodigde SKUs in elke zone te picken. Zodra alle SKUs van een order uit de verschillende zones gepickt zijn, kan door een sorteersysteem het order worden samengesteld (de Koster et al., 2007; Gu et al., 2010; Jane & Laih, 2005).

Bovenstaande methoden kunnen eveneens in combinatie worden toegepast (C. Petersen, 2009).

B Routering

De order picker start de pickroute in het depot, vervolgens loopt de picker doorheen het magazijn om SKUs te picken en beëindigt de route opnieuw in het depot. De order picker zal dan de gepickte SKUs deponeren in het depot en zal daar een nieuwe route ontvangen (K. J. Roodbergen & De Koster, 2001; Scholz et al., 2016). De route wordt zo bepaald dat de material handling kosten minimaal zijn (Gu et al., 2007). Indien meerdere items van een te picken SKU in voorraad zijn, dan moet het te picken items nog bepaald worden. Hiervoor bestaan meerdere keuzemogelijkheden, twee voorbeelden zijn de houdbaarheidsdatum en het FIFO (first-in-first-out) principe (de Koster et al., 2007). Daarnaast moet bij het bepalen van de route rekening gehouden worden met eventueel eenrichtingsverkeer in de pickgangen (K. J. Roodbergen & De Koster, 2001) en met de verplaatsingsrichting van het transportmiddel waarop de werknemer zich bevindt. Het

transportmiddel zal namelijk slechts in één richting (horizontaal of verticaal) kunnen bewegen of zal zich gelijktijdig in twee richtingen (schuin) kunnen bewegen (Engelbregt & Kruijer, 2009). In 'Tabel 5: ' worden een reeks voorbeeldmethoden om de route te bepalen besproken. De visuele voorbeelden in deze tabel geven een situatie weer waarbij tweerichtingsverkeer in de gangen mogelijk is en het transportmiddel zich slechts in één richting tegelijkertijd kan bewegen.

Aisly-by-aisle De order picker bezoekt elke pickgang waarin een hij/zij een SKU moet picken. 	**Traversal or S-shape** De order picker bezoekt elke sub-gang (d.w.z. deel van een pickgang in éénzelfde blok) met ten minste één picklocatie. 
Return De order picker bezoekt een pickgang om een SKU te picken en verlaat vervolgens de pickgang aan dezelfde zijde waar hij/zij binnen ging. 	**Largest gap** De order picker bezoekt een pickgang tot aan het begin van de 'largest gap' en keert terug om de pickgang aan hetzelfde einde te verlaten. Hierbij wordt de 'largest gap' gedefinieerd als de maximale afstand tussen twee aangrenzende picklocaties binnen één gangpad, of de maximale afstand tussen een gangpaduiteinde en een picklocatie. 
Exact algorithm Een optimale oplossing voor het routeringsprobleem wordt een traveling salesman problem genoemd. De oplossingsmethode van een dergelijk probleem behoort tot de toepassingen van de transportsector. Een variant van deze methode wordt besproken in '2.3 Gelijkaardige toepassingen'.	

Tabel 5: Overzicht met voorbeelden van routeringsmethoden (de Koster et al., 2007; K. J. Roodbergen & De Koster, 2001; Theys et al., 2010; van Gils et al., 2018).

Door van Gils et al. (2018) werd aangetoond dat de beslissingen over batching, zoning en routing elkaar significant beïnvloeden. Ondanks deze beslissingen los van elkaar werden besproken, moeten deze daarom samen worden genomen. Voor meer uitleg over de onderlinge relaties, zie van Gils et al. (2018).

2.2 Scattered storage

Zoals besproken in '1.1 Praktijkrelevantie' brengt de opkomst van de e-commerce sector nieuwe uitdagingen met zich mee. De vier belangrijkste uitdagingen in een B2C e-commerce magazijn zijn als volgt:

- het behandelen van kleine orders;
- het aanbieden van een breed assortiment;
- de variatie in werklading door bijvoorbeeld seizoensfactoren;
- klanten verwachten een snel leveringsproces (Weidinger et al., 2018).

Samengevat moeten kleine, dringende bestellingen uit een breed assortiment worden gepickt (Weidinger, 2018). Om in deze situaties lange picktijden te vermijden, heeft Weidinger Felix (2018) een nieuwe opslagstrategie, namelijk scattered storage, onderzocht. Bij deze strategie worden de binnenkomende ladingen uit de verpakking gehaald om vervolgens de individuele SKUs verspreid te stockeren over de rekken van het magazijn. Doordat de SKUs zich verspreid over het magazijn bevinden, is de kans groot dat een te picken SKU kortbij gelegen is en kan de picking tijd gereduceerd worden (Weidinger, 2018). Een order picker moet wel via een computersysteem begeleid worden, want het is onmogelijk om de dichtstbijzijnde locatie van een SKU vanbuiten te kennen. Verder is dit enkel interessant indien het voordeel van verkorte picktijden opweegt tegen de verhoogde inspanning om de rekken bij te vullen. Opslaglocaties komen namelijk sneller vrij doordat deze slechts gevuld zijn met een kleine hoeveelheid SKUs. Het bijvullen van de rekken kan echter gebeuren tijdens de daluren wanneer het order picking proces het minst wordt verstoord. Daarnaast kunnen de bijvulactiviteiten eveneens gebruikt worden om werknemers bekend te maken met het magazijn of om overtollig personeel bezig te houden (Weidinger & Boysen, 2018).

Een scattered storage strategie is echter niet altijd een interessante opslagstrategie om toe te passen. Een magazijn kan immers zowel online bestellingen moeten leveren als leveringen aan fysieke winkels. Het grote verschil tussen deze leveringen is de samenstelling van een order. Zo zal een online order minder vaak meerdere items van eenzelfde SKU bevatten, terwijl dat bij een levering naar een fysieke winkel wel het geval is (Weidinger et al., 2018). Indien een order bestaat uit meerdere items van eenzelfde SKU, dan kan een hogere spreiding van SKUs leiden tot langere picktijden. In dat geval moeten mogelijks meerdere opslaglocaties worden bezocht om alle items te verzamelen. Daartegenover, indien orders bijna nooit meerdere items van dezelfde SKU bevatten, zoals een online order, dan moet de spreiding van de SKUs gemaximaliseerd worden. In dat laatste geval kan de picktijd tot wel 55% verkort worden in vergelijking met een minder verspreide opslag (Weidinger, 2018). Volgens Weidinger et al. (2018) is het al interessant om scattered storage toe te passen indien 33% van de bestellingen via het internet worden geplaatst.

Om de scattered storage strategie verder in detail toe te lichten zal de volgende structuur gebruikt worden. Eerst zal de magazijn lay-out worden gekaderd waarin een scattered storage strategie in kan worden toegepast. Vervolgens wordt besproken hoe de SKUs optimaal verspreid kunnen worden over de opslaglocaties. Ten slotte wordt de optimalisatie van het order picking proces in een scattered storage magazijn toegelicht.

2.2.1 Lay-out van het magazijn

In de volgende secties zullen verschillende modellen worden besproken om de scattered storage strategie toe te passen. Om de lezer beter bekend te maken met de situatie waarin deze modellen kunnen worden toegepast, wordt een typerend voorbeeld van de lay-out van een B2C e-commerce magazijn besproken:

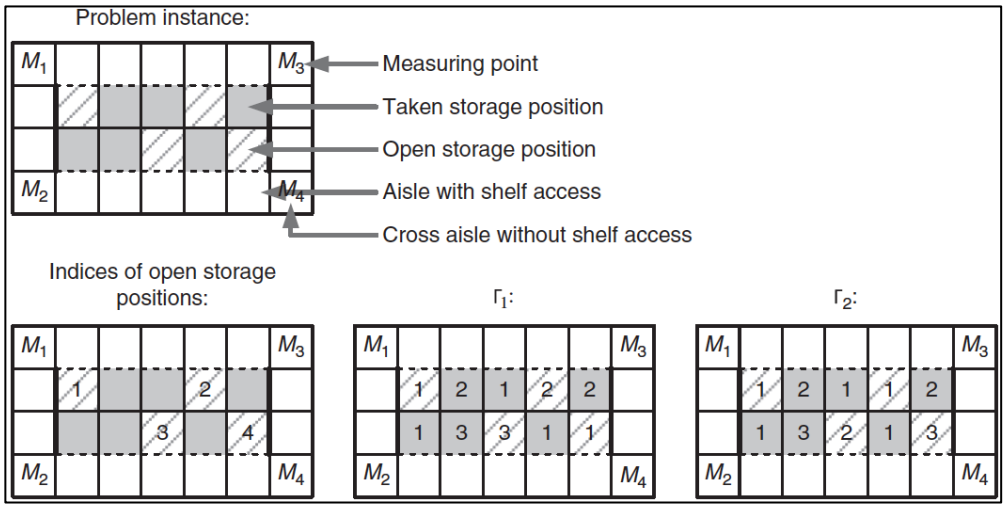
- De opslagrekken zijn op hoofdhoogte. Hierdoor zijn de SKUs in de rekken gemakkelijk bereikbaar voor order pickers (Weidinger et al., 2018; Weidinger & Boysen, 2018).
- De opslagrekken kunnen bestaan uit meerdere verdiepingen die opnieuw zijn opgedeeld in meerdere opslaglocaties (Weidinger et al., 2018; Weidinger & Boysen, 2018).
- De opslagrekken staan parallel t.o.v. elkaar. Aan elk uiteinde van de rekken bevindt zich een gangpad (Weidinger, 2018; Weidinger et al., 2018; Weidinger & Boysen, 2018).
- Het magazijn beschikt over één depot (Weidinger, 2018). Om de toegankelijkheid van het depot te verhogen, kan eventueel een lopende band systeem worden toegepast. Hierbij is het depot toegankelijk vanop meerdere locaties in het magazijn (Weidinger et al., 2018; Weidinger & Boysen, 2018). Deze strategie wordt 'decentralised depositing' genoemd (de Koster et al., 2007).
- De order picker kan eventueel over een pick-kar beschikken. De pick-kar bestaat uit bakken om meerdere orders gelijktijdig te verzamelen. Doordat de orders in een scattered storage magazijn klein zijn, kan een kleine en beweegbare pick-kar gebruikt worden (Weidinger et al., 2018; Weidinger & Boysen, 2018).
- De picker maakt gebruik van een een handscanner om zich volgens een vooraf bepaalde route doorheen het magazijn te verplaatsen. Daarnaast wordt de handscanner eveneens gebruikt om SKUs in of uit stock te plaatsen in het systeem. Een pick-by-voice systeem is een alternatief op een handscanner (Weidinger et al., 2018; Weidinger & Boysen, 2018).
- Vaak is de opslagafdeling in een e-commerce magazijn verder opgedeeld in zones. Dat is voordelig voor de modellen die later zullen besproken worden, want het zoekgebied van een zone is namelijk kleiner dan het volledige opslaggebied (van Gils et al., 2016).

2.2.2 Stockering

De scattered storage strategie is gekenmerkt door een spreiding van SKUs over het magazijn. Hierdoor zal een order picker zich nooit ver moeten verplaatsen om een te picken SKU te bereiken. In de meeste magazijnen wordt deze spreiding bepaald a.d.h.v. een willekeurig stockeringsproces. Deze strategie heeft als gevolg dat een gelijke verdeling eerder een toeval is. In de paper van Weidinger & Boysen (2018) wordt het vraagstuk om de SKUs optimaal te spreiden een scattered storage assignment (SSA) probleem genoemd. De oplossing van het probleem wijst een aantal items van één of meerdere SKUs toe aan vrije opslaglocaties in een deels gevuld magazijn. Het probleem maakt daarvoor gebruik van meetpunten. Dat zijn vooraf bepaalde locaties in het magazijn, bijvoorbeeld de kruising tussen een gangpad en een pickgang, van waaruit de afstanden tot het dichtstbijzijnde item van elke SKU kunnen worden bepaald. Het doel van het SSA probleem is bijgevolg om de doelfunctie $F(\Gamma) = \sum_{i \in I} w_i \cdot \max_{\tau \in D} \left\{ \min\{d_{\tau,i}(\Gamma);\ \delta_{\tau,i}\} \right\}$ te minimaliseren, met

- Γ een oplossing voor het invullen van alle lege opslaglocaties,
- w_i een wegingsfactor specifiek voor SKU i, en
- $\delta_{\tau,i}$ de afstand van meetpunt τ naar het dichtstbijzijnde item van SKU i.

Deze doelfunctie tracht voor elke SKU i de grootste afstand naar het dichtstbijzijnde item van de SKU over alle meetpunten te minimaliseren. Een visuele weergave van het probleem is zichtbaar in 'Figuur 6'. Beschouw hieruit Γ_1 als een oplossing van het probleem waarbij SKU 1 t.e.m. 3 willekeurig over de vrije opslaglocaties werden verdeeld en beschouw SKU 2 als de huidige SKU om de doelfunctie $F(\Gamma_1)$ te berekenen. Eerst wordt vanuit elk meetpunt berekent hoe ver de afstand is tot het dichtstbijzijnde item van SKU 2. Deze waarden bedragen 2, 5, 1, 2 voor respectievelijk M_1, M_2, M_3 en M_4. Het maximum van deze waarden bedraagt 5 en bijgevolg wordt deze waarde vermenigvuldigd met het gewicht w_2 om de doelfunctiewaarde te bekomen.



Figuur 6: Dit is een visueel voorbeeld van het scattered storage opslagprobleem (Weidinger & Boysen, 2018).

Merk op dat het SSA probleem wordt opgelost in een niet-leeg magazijn d.w.z. een deel van de opslaglocaties zijn reeds ingevuld met items van SKUs. Hierdoor kan $\delta_{\tau,i}$ de afstand zijn tussen meetpunt τ naar een opslaglocatie die reeds ingevuld was, ofwel de afstand tussen meetpunt τ naar een open opslaglocatie S die zonet werd ingevuld met een item van SKU i. Het SSA probleem kan optimaal worden opgelost via een mixed-integer probeem (MIP) model. Voor een gedetailleerde bespreking van het SSA-MIP model, wordt verwezen naar de paper van Weidinger & Boysen (2018).

Volgens een wiskundig bewijs werd aangetoond dat bovenstaand MIP-model sterk NP-hard is. Dat wil zeggen dat het een complex probleem is en daardoor mogelijks moeilijk op te lossen is bij grote datasets. Daarom ontwikkelde Weidinger & Boysen (2018) een heuristieke oplossingsmethode voor het SSA probleem gebaseerd op het p-center probleem om de SKUs te spreiden. In een p-center probleem moet de gewogen maximale afstand tot een bepaalde groep klanten worden geminimaliseerd door exact p faciliteiten te lokaliseren. Het algemene idee van het p-center probleem wordt vervolgens toegepast op het scattered storage opslagvraagstuk. Het doel is, net zoals in het MIP-model, om voor elke SKU de grootste afstand naar het dichtstbijzijnde item van de SKU over alle meetpunten te minimaliseren. De aanpak is echter verschillend door het gebruik van een

afstandsvector. Deze vector bevat voor elke SKU een maximaal af te leggen afstand van elk meetpunt tot het dichtstbijzijnde item van de SKU. Voor elk meetpunt wordt bijgehouden indien aan de maximale afstandswaarde is voldaan. Indien dit niet het geval is, moet die SKU worden toegewezen aan één van de vrije locaties binnen de maximaal af te leggen afstand t.o.v. het huidige meetpunt. Deze methode wordt door het gebruik van de afstandsvector het SSA probleem met maximum distances (SSA-MD) genoemd (Weidinger & Boysen, 2018).

Met bovenstaand uitgelegde SSA-MD methode kan een oplossing van het SSA probleem gevonden worden. Het is dan echter nog niet zeker dat dit een goede oplossing is. Daarom zal een adapted binary search (ABS) heuristiek gebruikt worden die bovenstaande SSA-MD methode meermaals aanroept. Het ABS is op te delen in twee onderdelen. Het eerste deel van de ABS start door een boven- en ondergrens voor de maximale afstand tot een item van elke SKU te bepalen. De bovengrens wordt bepaald door de huidig gevulde opslaglocaties in het magazijn. De maximaal af te leggen afstand tot een SKU kan namelijk niet slechter zijn dan de maximale afstand in de huidige oplossing. Om de ondergrens te bepalen wordt het SSA-MD probleem opgelost. Het probleem wordt ingevuld met zo laag mogelijke maximale afstanden tot de SKUs en bijgevolg een beperkte verzameling van opslaglocaties die hieraan voldoen. De maximale afstanden worden vervolgens geleidelijk verhoogd totdat een haalbare oplossing voor het SSA-MD probleem wordt gevonden. Deze maximale afstanden worden de ondergrens. Met alle waarden van maximale afstanden tussen deze twee grenzen kan een zoekdomein worden bepaald. Dat domein bevat alle mogelijke afstandsvectoren en de bijbehorende doelwaarde. De gevonden vectoren worden tweemaal gesorteerd, namelijk o.b.v. de som van de afstanden in de vector en vervolgens o.b.v. dalende doelfunctiewaarden. Een voorbeeld van een zoekdomein is zichtbaar in '*Figuur 7*'. Het eerste deel van de ABS heuristiek sluit af door een mogelijke oplossing voor het opslagvraagstuk te genereren via een greedy heuristiek[8] en vervolgens deze oplossing te verbeteren via een lokale zoekprocedure.

Sum:	Maximum distance combinations and corresponding objective values ($\bar{\gamma}_1, \bar{\gamma}_2, \bar{\gamma}_3 \mid Z$):					
15	(3, 5, 7 | 20)					
14	(3, 4, 7 | 18)	(2, 5, 7 | 19)	(3, 5, 6 | 19)			
13	(3, 3, 7 | 16)	(2, 4, 7 | 17)	(3, 4, 6 | 17)	(2, 5, 6 | 18)	(3, 5, 5 | 18)	
12	(2, 3, 7 | 15)	(3, 3, 6 | 15)	(2, 4, 6 | 16)	(3, 4, 5 | 16)	(2, 5, 5 | 17)	(3, 5, 4 | 17)
11	(2, 3, 6 | 14)	(3, 3, 5 | 14)	(2, 4, 5 | 15)	(3, 4, 4 | 15)	(2, 5, 4 | 16)	
10	(2, 3, 5 | 13)	(3, 3, 4 | 13)	(2, 4, 4 | 14)			
9	(2, 3, 4 | 12)					

Figuur 7: Dit is een visueel voorbeeld van een mogelijk zoekdomein via de ABS heuristiek (Weidinger & Boysen, 2018).

[8] Deze heuristiek bepaalt snel een haalbare startoplossing voor het SSA door via een simpele gedachtegang de opslaglocaties één voor één toe te wijzen. Voor de exacte werking wordt verwezen naar de paper van Weidinger & Boysen (2018).

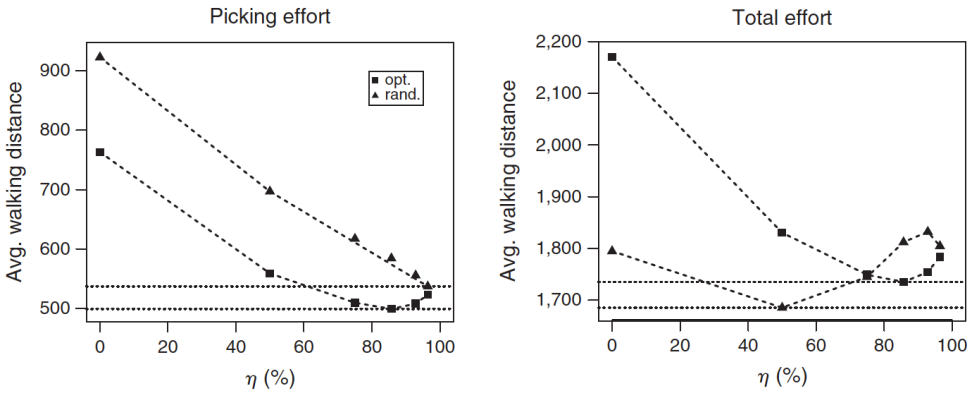
De tweede stap van de ABS-heuristiek bestaat uit een aantal iteraties. Tijdens elke iteratie wordt een even groot deel van het zoekdomein onderzocht waarbij de nadruk ligt op mogelijke oplossingen die dichtbij de huidig gevonden oplossing liggen. De selectie uit het zoekdomein wordt getest op haalbaarheid. Indien een mogelijke oplossing haalbaar blijkt en een betere oplossing voor het vraagstuk geeft, dan wordt de huidig gevonden oplossing overschreven en wordt de bovengrens aangepast. Het zoekproces stopt indien een vastgelegd aantal iteraties werd bereikt of indien de gevonden oplossing gelijk is aan de ondergrens.

Na het definiëren van een oplossing voor het SSA probleem werden de gevonden methodes getest. Eerst werd de werking van het SSA-MD probleem bestudeerd door de relatie tussen de berekeningstijd en drie factoren te onderzoeken. De eerste twee factoren zijn het aantal open opslaglocaties en het aantal gedefinieerde meetpunten. Beide factoren hebben een positieve, convexe relatie met de nodige berekeningstijd d.w.z. dat de berekeningstijd zal stijgen naarmate de factoren stijgen. Dat komt doordat deze factoren de input data van het SSA-MD probleem vergroten. De derde factor, de vullingsgraad van het magazijn, heeft een negatief convexe relatie met de berekeningstijd. Hoe lager de vullingsgraad, hoe hoger het aantal opslaglocaties die opgevuld moeten worden met een SKU en bijgevolg is het het aantal beslissingsvariabelen hoger. Eventuele opslagrestricties voor bepaalde SKUs versnellen dan weer de berekening doordat minder vrije opslaglocaties in aanmerking komen per SKU. Voor grote datasets (1.000 vrije opslaglocaties) vindt de methode voor het oplossen van het SSA-MD probleem een oplossing in ongeveer 1,1 seconde.

Om de werking van de ABS-heuristiek te testen werd deze vergeleken met twee andere oplossingsmethoden. Dat zijn het oplossen van het SSA-MIP d.m.v. Gurobi (een oplossingsprogramma) en het oplossen van het SSA probleem d.m.v. een referentiepunt (een greedy heuristiek gevolgd door een beperkte lokale zoekfase). In het geval van kleine datasets vindt Gurobi in 98,75% de optimale oplossing. Echter bij het gebruik van grote datasets vindt Gurobi moeilijk een optimale oplossing. De ABS-heuristiek vindt bij de kleine datasets een gemiddelde kloof van 2,39% tussen de gevonden en de optimale oplossing. Bij grote datasets vindt de ABS-heuristiek de beste oplossingen (t.o.v. de gevonden oplossingen door Gurobi en het referentiepunt). Weidinger & Boysen (2018) besloten dat de ABS-heuristiek geschikt is om te gebruiken indien het magazijn kan worden opgedeeld in subproblemen.

Het doel van een scattered storage strategie is dat pickers kortere afstanden moeten afleggen tijdens het pickingproces en bestellingen bijgevolg sneller worden samengesteld. Dat doel wordt door Weidinger & Boysen (2018) benaderd d.m.v. de formulering van het SSA probleem die SKUs optimaal tracht te spreiden over de opslaglocaties. Deze benadering wordt getest door de oplossingen gevonden door de ABS-heuristiek te vergelijken met de pickroutes gevonden in een random storage strategie. Daaruit blijkt dat het toepassen van de ABS-heuristiek (opt.) over het algemeen kortere pickroutes oplevert, zoals zichtbaar in ‘Figuur 8’. Uit deze figuur valt eveneens af te leiden dat het gemiddelde verschil tussen de gemiddelde lengte van de pickroutes tussen de twee strategiën steeds kleiner wordt naarmate de vullingsgraad van het magazijn toeneemt. Indien de twee strategiën worden vergeleken o.b.v. de totale inspanning (m.a.w. de wandelafstand tijdens bijvul- en pickactiviteiten), dan blijkt een spreiding via de ABS-heuristiek niet altijd voordelig te zijn. Op ‘Figuur

8’ is zichtbaar dat het toepassen van de ABS-heuristiek pas optimaal is op vlak van totale inspanning bij een vullingsgraad van 86%. De random assignment strategie daarentegen is optimaal bij een magazijn dat slechts voor de helft gevuld is. Op deze minimale punten bedraagt de kloof tussen de totale inspanning slechts 2,87%.



Figuur 8: Een overzichtsfiguur van de invloed van de vullingsgraad op respectievelijk de gemiddelde pickroutelengte en de gemiddelde totale routelengte (Weidinger & Boysen, 2018).

Om te bepalen wanneer het bijvulproces best kan starten, is de totale inspanning niet de enige factor waarbij rekening moet worden gehouden. Het startmoment is namelijk eveneens in afweging met de volgende redenering. Indien het bijvulproces continu gebeurt, dan zijn slechts enkele opslaglocaties vrij om de SKUs over te verdelen. Daarentegen, indien het bijvulproces wordt uitgesteld, dan komen steeds meer opslaglocaties vrij om opnieuw optimaal over te verdelen. Het kan nadelig zijn dat een SKU daardoor op minder opslaglocaties beschikbaar is. De order picker zal mogelijks een langere afstand moeten afleggen om de gewenste SKU te picken. Deze afweging blijkt in een scattered storage magazijn uit onderzoek van Weidinger & Boysen (2018) optimaal te worden opgelost wanneer 15% van de opslaglocaties leeg zijn. Voor een random storage strategie daarentegen is dit percentage moeilijker vast te leggen. Om de afgelegde afstand van de orderpicker beperkt te houden, is een continu bijvulproces de beste keuze. Daarentegen is het bijvulproces pas efficiënt als het magazijn tamelijk leeg is (Weidinger, 2018).

Ten slotte wordt de invloed van het aantal meetpunten op de routelengte onderzocht. Bij een hoger aantal meetpunten wordt steeds een kortere gemiddelde routelengte gevonden. Deze relatie vlakt echter snel af, namelijk na twintig meetpunten wordt slechts nog marginale verbeteringen van de gemiddelde routelengte gevonden. Uit dit resultaat kan worden afgeleid dat relatief weinig meetpunten al voldoende kunnen zijn om tot redelijk goede oplossingen te leiden (Weidinger, 2018).

2.2.3 Picker-routing

Picker-routing in een scattered storage magazijn is een gecombineerd probleem bestaande uit twee subproblemen. Eerst moet worden bepaald van welke opslaglocatie een SKU gepickt zal worden en vervolgens moet een route tussen de gekozen opslaglocaties worden samengesteld (Weidinger, 2018). Momenteel werd het picker-routing vraagstuk slechts tweemaal onderzocht, namelijk door Weidinger (2018) en Weidinger et al. (2018). De tweede paper bouwt verder op de resultaten van de eerste paper.

2.2.3.1 Het eerste onderzoek

Het picker-routing probleem werd door Weidinger (2018) onderzocht in de volgende situatie:

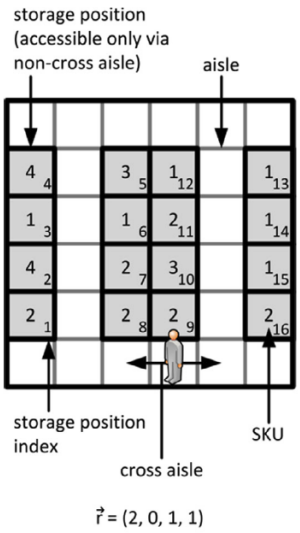
- De vorm van het magazijn is rechthoekig.
- De pick-tour begint en eindigt bij het depot.
- Een order picker pickt slechts één order of één vooraf vastgelegde batch orders gelijktijdig.
- De order picker verzamelt de te picken SKUs uit een verzameling van opslaglocaties met de desbetreffende SKUs.

Op basis van bovenstaande situatie werd een MIP-model (mixed-integer-problem) opgesteld om het picker-routing vraagstuk optimaal op te lossen. Optimaal wilt hier zeggen dat de pickroute een minimale afstand bedraagt. Volgens een wiskundig bewijs werd aangetoond dat het MIP-model voor deze situatie sterk NP-hard is. Daarom werden drie heuristieken ontwikkeld die het picker-routing probleem kunnen oplossen voor zowel grote als kleine datasets. De drie heuristieken kunnen elk worden opgedeeld in twee subproblemen, namelijk de selectie van locaties en de routering tussen de geselecteerde locaties. Concreet krijgt elke te beschouwen opslaglocatie een score o.b.v. de geschatte afstand die een picker bijkomend moet afleggen om die positie te bereiken. Hoe lager de waarde van prioriteit, hoe kleiner bijgevolg de additionele af te leggen afstand. Het selectievraagstuk is waar de drie heuristieken verschillen van elkaar. Weidinger (2018) ontwikkelde namelijk drie prioriteitsregels. De eerste prioriteitsregel, SinglePosition genaamd, wordt berekend voor een gegeven opslaglocatie j. Vanuit deze locatie wordt berekend hoeveel extra afstand de picker moet afleggen om opslaglocatie i te bereiken. De tweede en derde prioriteitsregel beschikken beide over een gelijkaardige gedachtegang. Bij de SinglePosition prioriteitsregel kreeg de afstand tussen twee opslaglocaties een score. Bij de komende twee prioriteitsregels wordt de te beschouwen locatie vergeleken met de tot nu geselecteerde locaties. Meer concreet evalueert de tweede regel, namelijk MinMax, de maximale afstand tussen de geselecteerde posities en de beschouwde positie en de derde regel, namelijk MinMin, evalueert de minimale afstand tussen de geselecteerde posities en de beschouwde positie. Alle prioriteitsregels beschikken eveneens over eenzelfde tiebreaker om een keuze mogelijk te maken tussen twee opslaglocaties met een gelijke score. Het is namelijk zo dat de opslaglocatie met de grootste voorraad zal gekozen worden doordat deze een lagere waarde na de komma krijgen toegewezen via de term $\frac{\bar{q}-q_i}{\bar{q}}$. Na de selectie van de opslaglocaties via één van de prioriteitsregels moet de kortste route tussen de locaties nog worden bepaald. Dat tweede subprobleem kan optimaal worden opgelost voor een magazijn met een eenvoudige lay-out zoals vastgelegd in de assumpties. In 'Tabel 6' wordt de werking van de heuristieken samengevat (Weidinger, 2018).

	SinglePosition	**MinMax en MinMin**
Notatie	c_{ij} = de afstand tussen opslaglocatie i en opslaglocatie j q_i = het aantal items in opslaglocatie i $\bar{q}$ = het maximale aantal items per opslaglocatie $\tilde{N}$ = de verzameling van reeds geselecteerde opslaglocaties	

Hoe starten?	Sorteer de te picken SKUs o.b.v. het aantal beschikbare opslaglocaties om het SKU van te picken. (Indien meerdere SKUs over even veel opslaglocaties beschikken, dan worden de SKU gesorteerd op hun index.) Voer vervolgens onderstaand stappenplan uit voor elke opslaglocatie van de SKU met het laagste aantal opslaglocaties.	Sorteer de te picken SKUs o.b.v. het aantal beschikbare opslaglocaties om het SKU van te picken. (Indien meerdere SKUs over even veel opslaglocaties beschikken, dan worden de SKU gesorteerd op hun index.) Voer vervolgens onderstaand stappenplan uit voor elk te picken SKU en start bij de SKU met het laagste aantal opslaglocaties.
Stappenplan	1) Stel j gelijk aan de index van de te beschouwen opslaglocatie van de SKU met het minste aantal opslaglocaties. 2) Bereken voor elke opslaglocatie van alle te picken SKUs de prioriteitswaarden. Stel i hier gelijk aan de te beschouwen opslaglocatie van een te picken SKU. $$P_{SP}^{j}(i) = c_{ij} + \frac{\bar{q} - q_i}{\bar{q}}$$ 3) Selecteer opslaglocaties met de laagste prioriteitswaarden totdat aan de vraag van elk SKU voldaan is. 4) Deselecteer opslaglocaties van een SKU indien de vraag voldaan blijft. Start hierbij vanaf de hoogste prioriteitswaarde en ga door totdat de laagste prioriteitswaarde wordt bereikt. 5) Voeg de geselecteerde opslaglocaties toe aan $\tilde{N}$ en bereken de kortste route voor $\tilde{N}$. Herhaal het stappenplan voor elke opslaglocatie van de SKU met het laagste aantal opslaglocaties. Selecteer ten slotte $\tilde{N}$ met de kortste bijbehorende route.	1) Bereken de prioriteitswaarden voor alle locaties van de huidige SKU. MinMax: $P_{max}(i) = \max_{j \in \tilde{N}}\{c_{ij}\} + \frac{\bar{q} - q_i}{\bar{q}}$ Of MinMin: $P_{min}(i) = \min_{j \in \tilde{N}}\{c_{ij}\} + \frac{\bar{q} - q_i}{\bar{q}}$ 2) Selecteer opslaglocaties met de laagste prioriteitswaarden totdat aan de vraag van het SKU voldaan is. 3) Deselecteer opslaglocaties van een SKU indien de vraag voldaan blijft. Start hierbij vanaf de hoogste prioriteitswaarde en ga door totdat de laagste wordt bereikt. 4) Voeg de geselecteerde opslaglocatie(s) toe aan $\tilde{N}$. Herhaal het stappenplan voor de volgende SKU met het minste aantal opslaglocaties. Bereken ten slotte de route voor de opslaglocaties in $\tilde{N}$.

Tabel 6: Een uitwerking van de drie heuristieken (Weidinger, 2018)



SinglePosition	MinMax	MinMin
Iteration 1: j = 5 ⇦ $P_{SP}^{5}(3) = 1$ ⇦ $P_{SP}^{5}(6) = 1$ ⇦ $P_{SP}^{5}(12) = 5$ $P_{SP}^{5}(13) = 5$ $P_{SP}^{5}(14) = 6$ $P_{SP}^{5}(15) = 7$ SKU 1 $P_{SP}^{5}(2) = 2$ SKU 4 $P_{SP}^{5}(4) = 0$ ⇦ $\tilde{N} = \{0, 3, 4, 5, 6\}$ → tour length: 12 Iteration 2: j = 10 ⇦ $P_{SP}^{10}(3) = 8$ $P_{SP}^{10}(6) = 8$ $P_{SP}^{10}(12) = 2$ $P_{SP}^{10}(13) = 2$ $P_{SP}^{10}(14) = 1$ ⇦ $P_{SP}^{10}(15) = 0$ ⇦ SKU 1 $P_{SP}^{10}(2) = 7$ ⇦ SKU 4 $P_{SP}^{10}(4) = 7$ $\tilde{N} = \{0, 2, 10, 14, 15\}$ → tour length: 16	Iteration 1 (SKU 3): $\tilde{N} = \{0\}$ $P_{max}(5) = 6$ $P_{max}(10) = 3$ ⇦ Iteration 2 (SKU 4): $\tilde{N} = \{0, 10\}$ $P_{max}(2) = 7$ ⇦ $P_{max}(4) = 7$ Iteration 3 (SKU 1): $\tilde{N} = \{0, 2, 10\}$ $P_{max}(3) = 8$ $P_{max}(6) = 8$ $P_{max}(12) = 7$ ⇦ $P_{max}(13) = 7$ ⇦ $P_{max}(14) = 8$ $P_{max}(15) = 7$ $\tilde{N} = \{0, 2, 10, 12, 13\}$ → tour length: 16	Iteration 1 (SKU 3): $\tilde{N} = \{0\}$ $P_{min}(5) = 6$ $P_{min}(10) = 3$ ⇦ Iteration 2 (SKU 4): $\tilde{N} = \{0, 10\}$ $P_{min}(2) = 4$ ⇦ $P_{min}(4) = 6$ Iteration 3 (SKU 1): $\tilde{N} = \{0, 2, 10\}$ $P_{min}(3) = 1$ ⇦ $P_{min}(6) = 1$ $P_{min}(12) = 2$ $P_{min}(13) = 2$ $P_{min}(14) = 1$ $P_{min}(15) = 0$ ⇦ $\tilde{N} = \{0, 2, 3, 10, 15\}$ → tour length: 16

Figuur 9: Voorbeelden van de drie heuristieken (Weidinger, 2018).

Om de werking van de drie prioriteitswaarden verder te duiden, wordt een voorbeeld van elke methode kort toegelicht a.d.h.v. 'Figuur 9'. De vector r bevat de gevraagde SKUs, namelijk twee eenheden van SKU 1, nul eenheden van SKU 2, één eenheid van SKU 3 en één eenheid SKU 4. De SinglePosition methode gaat van start door de SKU met het laagste aantal opslaglocaties te bepalen. In dit geval hebben zowel SKU 3 en SKU 4 twee opslaglocaties, bijgevolg wordt SKU 3 door de lagere index gekozen. SKU 3 is op twee opslaglocaties (5 en 10) beschikbaar, daarom zullen twee iteraties nodig zijn om de SinglePosition methode op te lossen. Tijdens de eerste iteratie is opslaglocatie j gelijk aan 5. Vervolgens worden voor alle opslaglocaties van de te picken SKUs de prioriteitswaarden berekend. De locaties worden ge(de)selecteerd volgens de regels in 'Tabel 6'. De route van geselecteerde opslaglocaties kan worden berekend en de tweede iteratie verloopt identiek aan de eerste iteratie. De gevonden oplossing van de eerste iteratie wordt gekozen doordat deze korter is. Door de sterk gelijkaardige werking van de MinMax en MinMin methoden wordt enkel de MinMax methode besproken. De methode gaat van start op dezelfde manier als de SinglePosition methode, bijgevolg wordt SKU 3 geselecteerd voor de eerste iteratie. SKU 3 beschikt over twee opslaglocaties waarvoor de prioriteitswaarde wordt berekend. Weeral volgens dezelfde (de)selectieregels wordt een opslaglocatie aan de route toegevoegd. De volgende iteratie gaat van start door de volgende SKU met het minste aantal opslaglocaties te beschouwen en opnieuw dezelfde stappen uit te voeren als in de eerste iteratie. Al wordt nu de prioriteitswaarde berekend door de beschouwde opslaglocatie te vergelijken met de twee locaties (0 en 10) die al in de route zijn opgenomen. Iteraties blijven volgen totdat voor alle te picken SKUs opslaglocaties geselecteerd zijn. Ten slotte wordt de route voor de geselecteerde locaties bepaald (Weidinger, 2018).

De drie heuristieken werden getest op zowel kleine als grote datasets. In de kleine en grote datasets zijn respectievelijk 8, 12 of 16 en 400 of 600 SKUs aanwezig in het magazijn. De denkbeeldige magazijnen werden opgevuld door eerst elke SKU al op één locatie te leggen. Vervolgens werden de overige opslaglocaties opgevuld met de logica van een ABC-classificatie. Voor elke opslaglocatie werd bepaald met welke klasse deze zou gevuld worden en vervolgens werd een willekeurige SKU uit die klasse geselecteerd. De kans op A-klasse was 80%, B-klasse 15% en C-klasse 5%. Voor elk magazijn met een verschillend aantal SKUs werden de locaties allemaal opgevuld met één item, ofwel kregen de opslaglocaties een aantal items willekeurig geselecteerd tussen één en drie. Vervolgens werden voor alle verkregen situaties picklijsten opgesteld. De picklijst werd bepaald door de aanwezige SKUs in het magazijn in een willekeurige volgorde te plaatsen. De lengte van de picklijst kon opnieuw variëren tussen drie, vijf of zeven orderlijnen. Voor elke bekomen situatie werden opnieuw tien voorbeelden aangemaakt. Daardoor bestaan de kleine en de grote datasets uit respectievelijk 60 en 40 voorbeeld magazijnen (Weidinger, 2018).

Om de oplossingsprestaties van de drie heuristieken te testen worden de voorbeeld magazijnen als data gebruikt. Als referentiepunt wordt de data eveneens opgelost met Gurobi. Twee verschillende tijdslimieten worden gehanteerd bij Gurobi, namelijk drie uur (Gurobi3) en vijf minuten (Gurobi5). Deze stellen respectievelijk een grondige zoektocht en een realistische tijdslimiet in praktijk voor. Bij het gebruik van kleine datasets vindt Gurobi3 in 91,9% van de testdata de optimale oplossing. Bijgevolg weten we in deze gevallen of één van de andere methoden eveneens de optimale oplossing heeft gevonden. Zo vindt Gurobi5 in 99% van de bekende optimale oplossingen eveneens een optimale oplossing. Bij de SinglePosition, MinMax en MinMin heuristieken worden respectievelijk 60,3%, 75,6% en 48,8% van de reeds bekende optimale oplossingen gevonden. De MinMax heuristiek vindt bijgevolg het vaakst de optimale oplossing. Daarentegen beschikt de SinglePosition heuristiek over de kleinste kloof tussen de gevonden en de optimale oplossing, namelijk een kloof van 6,85%. Indien van de drie heuristieken telkens de beste oplossing wordt gekozen (een gecombineerde methode), dan bedraagt de gemiddelde kloof slechts 0,5% en worden 94% van de optimale oplossingen gevonden.

Voor de grote datasets vindt Gurobi3 in geen enkel geval een optimale oplossing, maar wel altijd een haalbare oplossing. Dat betekent dat de methoden nu met elkaar moeten vergeleken worden door telkens de best gevonden oplossing als vergelijkingspunt te kiezen. Gurobi5 slaagt zelfs niet om altijd een haalbare oplossing te vinden. Onder de heuristieken presteert nu de MinMin het beste door in 39,2% van alle gevallen de beste oplossing te vinden. Bij de SinglePosition en MinMin heuristiek bedragen de percentages slechts 18,3% en 8,3%. Door de heuristieken opnieuw te combineren wordt het percentage verhoogt tot meer dan 60% met een gemiddelde kloof van 5,4%. Deze statistieken van de gecombineerde methode overtreffen Gurobi3 die een percentage haalt van 45% en een gemiddelde kloof van 22%. De gecombineerde methode behaalt bijgevolg zowel bij kleine als grote datasets goede resultaten (Weidinger, 2018).

Doordat de uitkomsten van het picker-routing vraagstuk bepaald worden door het SSA probleem wordt deze relatie door Weidinger (2018) onderzocht. Meer specifiek wordt de relatie tussen de verspreiding van SKUs in het magazijn en de diversiteit van SKUs op de picklijst onderzocht. Het

magazijn met 600 SKUs wordt meermaals opgesteld met verschillende mate van spreiding van de SKUs. Eveneens de picklijsten worden met verschillende mate van diversiteit opgesteld. Doordat de mate van diversiteit van de picklijst in praktijk gegeven is, moet nu bepaald worden wat de mate van spreiding in het magazijn moet zijn. Uit de resultaten onderscheiden zich dan drie scenario's. Het eerst scenario gaat over picklijsten met een hoge mate van diversiteit. In dat geval wordt de spreiding van SKUs over de opslaglocaties best gemaximaliseerd om de pickrondes aanzienlijk te verkorten. Dat komt doordat de kans dat verschillende SKUs dicht bij elkaar gestockeerd liggen aanzienlijk toeneemt. Het tweede scenario met picklijsten van een lage diversiteit stelt het omgekeerde. Het is dan verstandiger om een lage spreiding in het magazijn toe te passen. Bijgevolg liggen meerdere items van eenzelfde SKU korter bij elkaar waardoor het bezoeken van meerdere ver uiteenliggende locaties wordt vermeden. Ten slotte wordt het derde scenario gekenmerkt door een afwisseling tussen lage en hoge diversiteit op de picklijsten. Deze toepassing werd al kort gekaderd in de inleiding van '2.2 Scattered storage' door magazijnen te bespreken die zowel aan fysieke winkels leveren als aan de klanten van online winkels. Om in dat geval de pickrondes te verkorten, is het voordelig om kleine clusters van identieke SKUs verspreid over het magazijn aan te leggen.

2.2.3.2 Het tweede onderzoek

De paper van Weidinger et al. (2018) bouwt verder op de resultaten van voorgaand onderzoek door Weidinger (2018). De gevonden modellen worden aangepast om de realiteit beter te benaderen. Daardoor is dit de huidige situatie van het picker-routing probleem (de nieuwe of vernieuwde assumpties t.o.v. de vorige paper zijn onderlijnd):

- De vorm van het magazijn is rechthoekig.
- De pick-tour begint en eindigt in één van de toeganspunten tot het depot. Bijgevolg kan het start- en eindpunt van een pickroute een verschillend toeganspunt zijn.
- Een order picker maakt gebruik van een picker kar en kan daardoor meerdere orders of meerdere batches van orders gelijktijdig picken.
- De order picker verzamelt de te picken SKUs uit een verzameling van opslaglocaties met de desbetreffende SKUs.

Op basis van bovenstaande situatie werd een MIP-model opgesteld. Opnieuw werd aangetoond dat het MIP-model voor deze situatie sterk NP-hard is. Daarom werden twee heuristieken ontwikkeld die het picker-routing probleem oplossen bij grote datasets (Weidinger et al., 2018). De eerste heuristiek is gebaseerd op de nearest neighbour heuristic en de werking is als volgt:

1. Bepaal de orders in de te picken batch. Dat kan bijvoorbeeld gebeuren door de first-come-first-served (FCFS) methode toe te passen.
2. Voeg de dichtstbijzijnde opslaglocatie van een SKU toe aan de route totdat alle SKUs van de batch gepickt zijn en voeg vervolgens het dichtstbijzijnde depot toe aan de route.
3. Herhaal bovenstaande stappen tot alle orders zijn gepickt.

De tweede heuristiek is gebaseerd op een pool-based constructie heuristiek. Een constructie heuristiek is een heuristiek die deeloplossingen stapsgewijs uitbreidt. De pool-based constructie heuristiek bestaat uit twee niveaus. Het onderste niveau werkt deeloplossingen uit waarvan telkens

de beste wordt toegevoegd aan de pool van het bovenste niveau. Het bovenste niveau bepaalt opnieuw de orders die in de volgende iteratie op het onderste niveau in nieuwe of bestaande deeloplossingen uitgewerkt zullen worden. Deze niveaus worden doorlopen totdat alle deeloplossingen zijn voltooid en alle orders zijn gepland. Vervolgens wordt de beste oplossing van de pool gekozen als eindoplossing van de heuristiek. Om de deeloplossingen op het onderste niveau uit te werken, kunnen bijvoorbeeld de prioriteitsheuristieken uit Weidinger (2018) gebruikt worden om de opslaglocaties te selecteren en een gekozen routeringsmethode kan gebruikt worden om de route tussen de locaties te bepalen (Weidinger et al., 2018).

Zodra een mogelijke oplossing voor het picker-routing probleem is gegenereerd via één van de twee heuristieken, kan een iteratieve zoekprocedure worden gebruikt om de oplossing verder te verbeteren. Deze methode wordt een verbeteringsheuristiek genoemd en probeert subtours van de picktour uit de gegeven oplossing te verbeteren. Deze nieuwe alternatieve subtours moeten natuurlijk nog steeds aan dezelfde hoeveelheid SKUs voldoen als de originele picktour (Weidinger et al., 2018).

Symbol	Description	Small	Large
w	Number of aisles	2	40
h	Length of aisles [in #storage positions]	5	150
$\|S\|$	Number of SKUs	{3, 6, 9}	{2000, 3500, 5000}
$\|D\|$	Number of depots	{2, 4}	{26, 80}
$\|O\|$	Number of orders	3	10
o_{max}	Number of items per order	{1, 2}	{1, 2}
γ_{max}	Max. number of items stored per storage position	{1, 2}	{1, 4}
C	Capacity of picking cart	2	4

Figuur 10: Een overzicht van de mogelijke parameter waarden van zowel de kleine als de grote dataset (Weidinger et al., 2018).

Om de bekomen resultaten te testen maakte Weidinger et al. (2018) een onderscheid tussen kleine en grote datasets. Elke soort dataset werd gemaakt door elke combinatieset van de parameters in 'Figuur 10' twintig keren aan te maken als een voorbeeld magazijn. Daardoor ontstonden 480 kleine datasets en 480 grote datasets.

De prestaties van vijf verschillende oplossingsmethoden worden met elkaar vergeleken. De eerste methode is de standaardoplosser Gurobi die het MIP oplost met een maximale berekeningstijd van 30 minuten. De tweede en de derde methode zijn respectievelijk de nearest neighbour heuristiek (NN) en de pool-based constructie heuristiek (pool). Ten slotte, de vierde en vijfde methode zijn NN en pool aangevuld met een lokale zoekprocedure (deze worden later vermeld als NN+ en pool+). Bij het gebruiken van de kleine dataset vindt Gurobi in 477 van de 480 voorbeelden de bewezen optimale oplossing. Net zoals bij de eerste paper worden deze optimale oplossingen gebruikt om te weten of één van de andere methoden eveneens de optimale oplossing heeft gevonden. De pool+ methode vindt 96% van de optimale oplossingen met een kloof van 0,55% tussen de gevonden oplossing en de bewezen optimale oplossingen. Naast deze belovende resultaten heeft de pool+ methode een gemiddelde oplossingstijd van slechts een honderdste van een seconde. De NN+ methode vindt gemiddeld nog sneller een oplossing, maar de kloof bedraagt 22%. Bij het gebruik van de grote datasets vindt Gurobi geen haalbare oplossingen. Bijgevolg kunnen enkel de oplossingen tussen de

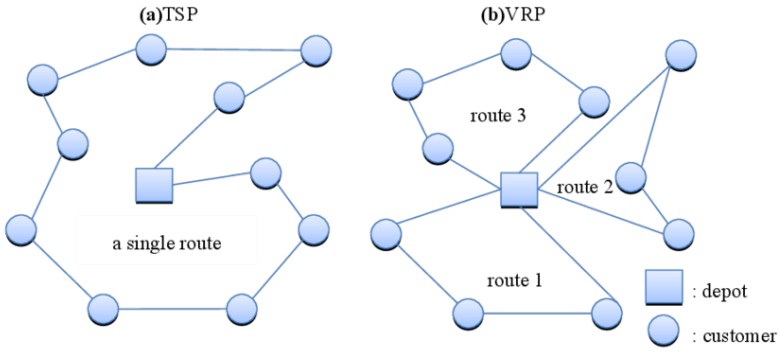
heuristieken worden vergeleken. Van de 480 voorbeelden vindt de pool+ methode 391 keer de beste oplossing en de resterende 89 beste oplossingen worden gevonden door NN+. In die 89 gevallen bedraagt de kloof tussen NN+ en pool+ ongeveer 1%. De pool+ heuristiek lijkt bijgevolg opnieuw de beste methode, maar de berekeningstijd is bij grote datasets wel toegenomen tot twintig seconden. Volgens Weidinger et al. (2018) is dit bij praktische toepassingen nog steeds acceptabel. Indien echter een minimale berekeningstijd wordt gewenst, dan presteert de nearest-neighbour heurstiek beter.

De lokale zoekprocedure verbetert gemiddeld gezien de resultaten van beide procedures voor zowel de kleine als de grote dataset. Bij de nearest neighbour heuristiek wordt de grootste verbetering gevonden doordat bij deze methode de initiële oplossing meestal van lagere kwaliteit is. Hoe meer iteraties worden uitgevoerd tijdens de zoekprocedure, hoe beter de gevonden oplossing. Het uitvoeren van bijvoorbeeld 150 iteraties duurt ongeveer 0,15 seconden. Bijgevolg neemt de lokale zoekprocedure slechts een beperkte tijd in beslag (Weidinger et al., 2018).

2.3 Gelijkaardige toepassingen

In dit onderdeel worden gelijkaardige toepassingen uit andere domeinen beschreven die kunnen helpen bij het oplossen van de onderzoeksvraag en bijbehorende deelvragen. De onderstaande toepassingen gaan beide over een probleemstuk betreffende picker-routing. Dat wil zeggen dat het probleemstuk in twee delen kan worden opgedeeld, namelijk de selectie van locaties en de routering tussen de geselecteerde locaties (Ghiani & Improta, 2000).

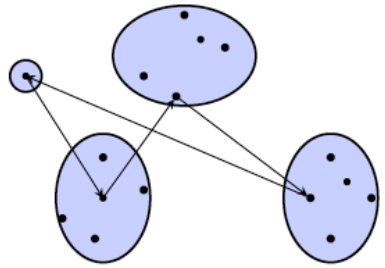
Eerst wordt een variant op het traveling salesman problem (TSP) besproken. Het TSP houdt rekening met een enkel voertuig dat meerdere klanten bezoekt voordat het terugkeert naar het depot en probeert de totale reistijd of -afstand van het voertuig te minimaliseren. Vervolgens wordt een variant op het vehicle routing problem (VRP) besproken. Het VRP is sterk gelijkend op het TSP, maar is verschillend doordat het VRP eveneens rekening houdt met beperkingen van de voertuigcapaciteit en met de vraag van de te bezoeken klanten. Daardoor kan de oplossing van het VRP meerdere routes bevatten, zoals zichtbaar op 'Figuur 11' (Baniasadi et al., 2020; Liu et al., 2014; Sundar & Rathinam, 2016).



Figuur 11: Een vergelijking tussen de oplossing van het TSP en het VRP voor dezelfde inputgegevens (Liu et al., 2014).

2.3.1 Generalized traveling salesman problem (GTSP)

Het generalized traveling salesman problem (GTSP) is een bekend combinatorisch optimalisatieprobleem dat gebaseerd is op een nog bekender probleem, namelijk het traveling salesman problem (TSP). In het GTSP worden, in tegenstelling tot in het TSP, de klanten opgedeeld in clusters en het voertuig moet elke cluster precies één keer bezoeken (Karapetyan & Gutin, 2012).



Figuur 12: Een voorbeeld van een oplossing van het GTSP (Baniasadi et al., 2020).

Een formele definitie van het GTSP wordt weergegeven als volgt: Gegeven is $G = (V, E)$ een niet-gerichte diagram met n knooppunten (V) waarvan de bogen (E) tussen de knooppunten beschikken over een niet-negatieve kost. Laat $V_1, \dots, V_p$ een deel zijn van V in p subsets genaamd clusters (d.w.z. $V = V_1 \cup V_2 \cup \dots \cup V_p$ en $V_l \cap V_k = \emptyset \; \forall \, l, k \in \{1, \dots, p\}$). De kost van een boog $e = \{i, j\} \in E$ bedraagt c_{ij}. Een haalbare oplossing is een route die precies één knooppunt in elke cluster $V_i, i \in \{1, \dots, p\}$ bezoekt. Het doel is om de goedkoopste route te vinden. Om deze optimale route te vinden, moeten twee beslissingen genomen worden: het kiezen van een knooppunt-subset $S \subseteq V$, zodat $|S \cap V_k| = 1, \forall \, k = 1, \dots, p$ en het vinden van een route tussen de geselecteerde knooppunten met minimale kosten. Het GTSP wordt symmetrisch genoemd als en slechts als de gelijkheid $c(i, j) = c(j, i)$ geldt voor elke $i, j \in V$, waarbij de kostenfunctie c bij de bogen van diagram G hoort. Indien $|V_i| = 1 \; \forall \, i$ (d.w.z. elke cluster bestaat slechts uit één knooppunt), dan reduceert het GTSP naar het TSP (Karapetyan & Gutin, 2012; Pintea et al., 2011; Smith & Imeson, 2017).

Het GTSP wordt eveneens toegepast in de logistieke context, namelijk order-picking in een magazijn met SKUs verspreid over verschillende opslaglocaties (Karapetyan & Gutin, 2012). Door enkele naamgevingen aan te passen kan de logistieke toepassing duidelijker worden:

- De klanten worden vervangen door SKUs. In een cluster zitten dan alle beschikbare opslaglocaties van een specifieke SKU.
- De wagen wordt vervangen door een order picker.
- De waardes verbonden aan de bogen worden afstanden tussen twee opslaglocaties of tussen een opslaglocatie en het depot. Het GTSP is symmetrisch, indien de afstanden in beide richtingen even lang zijn (Nalivajevs & Karapetyan, 2019).
- Het GTSP houdt nog geen rekening met de beschikbare voorraad op elke opslaglocatie.

Het GTSP is een grondig onderzocht probleem (Karapetyan & Gutin, 2012; Pintea et al., 2017; Smith & Imeson, 2017), daardoor zijn slechts een beperkt aantal voorbeelden van papers samengevat in onderstaande tabel.

Auteurs	Methode
Karapetyan & Gutin (2012)	De auteurs stelden dat veel heuristieke oplossingsmethoden voor het GTSP tot dan gebaseerd waren op een lokale zoekprocedure. Ze identificeerden de noodzaak om het bepalen van het lokale zoekgebied te verbeteren, aangepast aan het GTSP. Dat deden ze voor de volgende methoden: cluster optimization neigbhourhood, TSP-inspired neighbourhoods, fragment optimiztion neighbourhoods.
Pintea et al. (2017)	De auteurs ontwikkelden twee methoden om het GTSP op te lossen. De eerste methode was een exact exponentieel tijdalgoritme en de tweede methode een meta-heuristiek gebaseerd op het Ant Colony System (ACS) algoritme.
Smith & Imeson (2017)	De auteurs ontwikkelden een algoritme gebaseerd op de adaptive large neighbourhood search. Daarbij worden herhaaldelijk knooppunten toegevoegd in de route en verwijderd uit de route om de route te verbeteren.

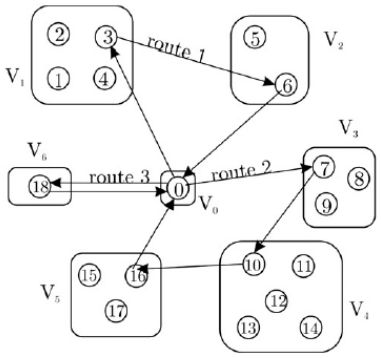
Tabel 7: Voorbeelden van papers over het GTSP.

Voor een oplossingsmethode van het GTSP in een logistieke context wordt verwezen naar de paper van Nalivajevs & Karapetyan (2019). In deze paper wordt een Conditional Markov Chain Search (CMCS) toegepast om het GTSP op te lossen in een order-picking omgeving. CMCS is een raamwerk ontworpen voor het automatisch genereren van optimalisatie-heuristieken. Het is een metaheuristiek gebaseerd op meerdere componenten. Elke component is een subroutine die een oplossing neemt en deze aanpast volgens de interne logica.

In de paper van Weidinger et al. (2018), over het picker-routing vraagstuk in een scattered storage magazijn, werden meerdere toeganspunten tot het depot gebruikt als assumptie. Daarentegen, in het GTSP wordt slechts één depot herkend. Indien meerdere depots worden toegevoegd aan diagram G, dan wordt het GTSP getransformeerd tot het generalized multiple depot traveling salesman problem (GMDTSP). Daardoor lijkt de context van het probleem meer op de assumpties gemaakt in de paper van Weidinger et al. (2018). Voor meer uitleg over het GMDTSP kan de paper van Sundar & Rathinam (2016) geconsulteerd worden.

2.3.2 Generalized vehicle routing problem (GVRP)

Het generalized vehicle routing problem (GVRP) is een uitbreiding van het klassieke vehicle routing problem (VRP). In het GVRP worden de klanten verdeeld in wederzijds exclusieve en uitputtende groepen (clusters). Het doel is om de optimale route te vinden voor elk van de voertuigen uit het wagenpark met vertrek- en eindpunt in het depot, die precies één klant uit elke groep bezoekt en dat rekening houdend met de capaciteitsbeperkingen van de wagens (Bektaş et al., 2011; Hà et al., 2014; P. C. Pop et al., 2013).



Figuur 13: Voorbeeld van een opgelost GVRP (P. C. Pop et al., 2013).

Het GVRP kan eveneens besproken worden aan de hand van een formele definitie. Gegeven is een gewogen volledig gerichte diagram $G = (V, E)$ (zie 'Figuur 13') met $V = \{0,1, \dots, n\}$ als de set knooppunten, $E = \{(i,j) \mid i,j \in V, i \neq j\}$ als de set bogen en een niet-negatieve kost c_{ij} geassocieerd met elke boog $(i,j) \in E$. De set knooppunten is onderverdeeld in $k+1$ elkaar uitsluitende niet-lege subsets, genaamd clusters, $V_0, V_1, \dots, V_k$ (d.w.z. $V = V_0 \cup V_1 \cup \dots \cup V_k$ en $V_l \cap V_p = \emptyset \; \forall \, l,p \in \{0,1, \dots, k\}$ en $l \neq p$). De cluster V_0 heeft slechts één knooppunt 0, dat het depot vertegenwoordigt, en de resterende n knooppunten die tot de resterende k clusters behoren, vertegenwoordigen de geografisch verspreide klanten. Elke klant heeft een niet-negatieve vraag q_i (met $q_i > 0$ voor $i = 1, \dots, k$ en $q_0 = 0$) die voldaan kan worden door elk van de knooppunten in de bijbehorende cluster. De beschikbare wagens m zijn identieke voertuigen elk met een capaciteit van Q. Het probleem wordt opgelost door het kiezen van een knooppunt-subset $S \subseteq V$, zodat $|S \cap V_i| = 1$, voor alle $i = 1, \dots, k$ en het vinden van een minimale kostenverzameling van routes waarbij wordt voldaan aan de capaciteitsbeperkingen en de routes starten en eindigen in het depot. Het GVRP reduceert zich tot het klassieke VRP wanneer elke cluster slechts uit één knooppunt bestaat en tot het generalized traveling salesman problem (GTSP) wanneer het wagenpark slechts uit één wagen met onbeperkte capaciteit bestaat (Baldacci et al., 2010; Bektaş et al., 2011; Ghiani & Improta, 2000; Hà et al., 2014; P. C. Pop et al., 2013).

Het bovenstaande besproken probleem is toepasbaar binnen het picker-routing vraagstuk door enkele naamgevingen aan te passen:

- De klanten worden vervangen door SKUs. In een cluster zitten dan alle beschikbare opslaglocaties van een specifieke SKU.
- Het wagenpark wordt vervangen door order pickers die beschikken over een pick-kar met een beperkte capaciteit.
- De waardes verbonden aan de bogen worden afstanden tussen twee opslaglocaties of tussen een opslaglocatie en het depot. Het GVRP is symmetrisch, indien de afstanden in beide richtingen even lang zijn.
- Het GVRP houdt nog geen rekening met de beschikbare voorraad op elke opslaglocatie.

Het probleem werd voor het eerst geïntroduceerd door Ghiani & Improta (2000). Ze stelden een oplossingsprocedure voor door het GVRP om te zetten in een Capacitated Arc Routing Probleem (CARP) waarvoor een exact algoritme en verschillende heuristieken reeds bestonden. Sindsdien werden enkele artikelen gepubliceerd die integer lineaire programmering formuleringen gebruiken om het GVRP op te lossen. In 2003 stelden Kara & Bektaş zo een oplossingsmethode voor met een

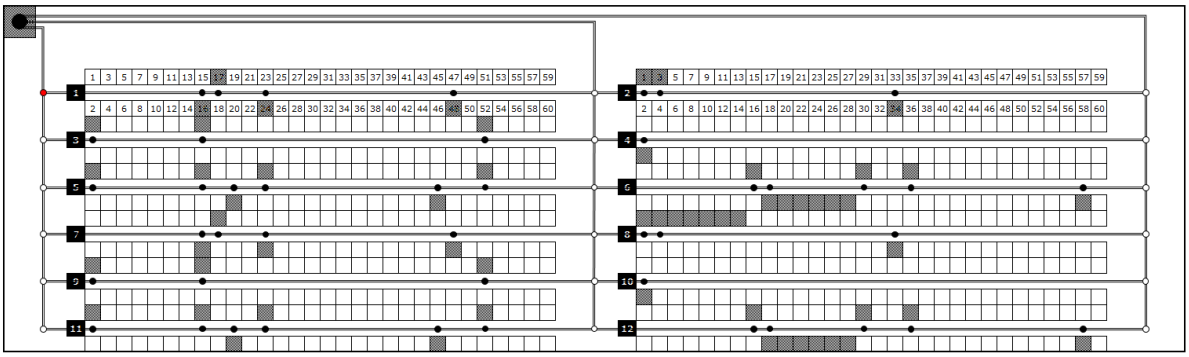
polynomiaal toenemend aantal binaire variabelen en beperkingen. Daarna introduceerde Bektaş et al. (2011) vier nieuwe integer lineaire programmeerformuleringen. Twee daarvan waren gebaseerd op multicommodity flow en de andere twee waren gebaseerd op exponentiële sets van ongelijkheden. P. Pop et al. bracht in 2011 twee nieuwe integer lineaire programmeermodellen uit. Daarnaast breidden de auteurs het GVRP uit met het geval waarin de hoekpunten van een cluster van elke tour aaneengesloten zijn. Twee jaar later werd het GVRP opgelost door Pop et al. (2013) d.m.v. een hybride heuristiek. Deze heuristiek combineert een genetisch algoritme met een lokaal-globale benadering van het probleem en met een lokale zoekprocedure. Hà et al. (2014) bekeek het GVRP vanuit een nieuwe hoek door het benodigde aantal voertuigen als de beslissingsvariabele te beschouwen. Vanuit deze invalshoek stelden de auteurs een exact algoritme en een metaheuristiek op voor het probleem. Bovenstaande papers vormen geen exhaustieve lijst van alle literatuur omtrent het GVRP-thema.

3 Praktijkstudie

De praktijkstudie gaat van start door de lay-out van het voorbeeld magazijn toe te lichten en de opstelling van de voorbeelddata te kaderen. Daarna wordt overgegaan tot de drie doelen van de praktijkstudie die elk gebaseerd zijn op het picker-routing probleem in een scattered storage magazijn. Het eerste doel is om de prioriteitsheuristieken uit de paper van Weidinger (2018), die de opslaglocatie-selectie uit het picker-routing probleem oplossen, te testen op een nieuwe dataset. De gevonden resultaten worden vergeleken met die van de paper. Het tweede doel is een analyse van het effect van batching op de resultaten van het picker-routing probleem. Eerst wordt de eenvoudige FIFO batching methode getest en vervolgens wordt een nieuwe batching methode voorgesteld die gebaseerd is op één van de prioriteitsheuristieken van Weidinger (2018). Ten slotte, het derde doel is om het effect te onderzoeken van de opslaglocatie-selectie in het picker-routing probleem op te lossen d.m.v. een parallelle/gelijktijdige beslissing voor alle te picken SKUs. Dat is in tegenstelling tot de sequentiële beslissing van de prioriteitsheuristieken van Weidinger (2018) waarbij de opslaglocatie-selectie SKU per SKU gebeurd.

3.1 Het voorbeeld magazijn

Het voorbeeld magazijn is gebaseerd op een reëel magazijn in Vlaanderen en is zichtbaar in 'Figuur 14'. Het magazijn bevat 720 opslaglocaties op grondniveau verdeeld over twaalf pickgangen. Het magazijn wordt verder onderveeld in twee blokken die elk zes van de pickgangen bevatten.



Figuur 14: De lay-out van het voorbeeld magazijn.

Elk van de 720 opslaglocaties kreeg een unieke index tussen 1 en 720 (OpslagLocatieID) zodat de locaties refereerbaar zijn tijdens de berekeningen. De nummering start linksboven en loopt vervolgens de gang af zoals zichtbaar in pickgang 1. Daarna gaat dezelfde manier van nummeren verder in subgang 2 t.e.m. subgang 12. Daarnaast laat het magazijn nog een andere nummering toe d.m.v. de combinatie van de gangindex (PickGangID) met de OpslagCelID. Hierbij stelt de OpslagCelID de nummering voor zoals in pickgang 1 en pickgang 2 op de figuur waarbij elke pickgang 60 uniek genummerde opslaglocaties bevat.

Verder gelden nog een aantal praktische richtlijnen in het voorbeeld magazijn:

- Pickers volgen altijd de lijnen op de figuur.
- Pickers kunnen vanaf deze lijnen in de subgangen producten uit beide opslagrekken/zijden picken.
- Picker mogen in twee richtingen een gang in en kunnen zich draaien in een gang.
- Om de afstanden in het magazijn weer te geven, wordt een afstandsmatrix gebruikt die de wandelafstand tussen alle locaties (inclusief het depot) voorstelt in deci-seconden.

3.2 Het opstellen van voorbeelddata

De voorbeelddata bestaat in totaal uit 30 tekstbestanden die picklijsten voorstellen. Deze bestanden zijn gebaseerd op dezelfde realistische data die gebruikt werd in de paper geschreven door van Gils et al. (2019) en is bijbehorend aan het voorbeeld magazijn. De tekstbestanden zijn onderverdeelbaar in drie groepen van telkens tien bestanden en bevatten de volgende kenmerken.

	Aantal bestanden	**Aantal orders per bestand**	**Maximaal aantal orderlijnen per order**
Groep 1	10	6	4
Groep 2	10	6	8
Groep 3	10	6	12

Tabel 8: Een overzicht van de kenmerken van de databestanden.

Elk tekstbestand bevat de volgende kolommen: OrderID, OrderLijnID, PickGangID, OpslagCelID en OpslagLocatieID. In onderstaand voorbeeldbestand bestaat het eerste order uit vier orderlijnen. Om de eerste orderlijn te picken moet in de tweede pickgang locatie 32 bezocht worden, oftwel de opslaglocatie met index 92 (indien de nummering van 1-720 wordt gebruikt).

OrderID	OrderlijnID	PickgangID	OpslagCelID	OpslaglocatieID
1	1	2	32	92
1	2	9	46	526
1	3	10	36	576
1	4	1	13	13
2	5	4	50	230
2	6	1	19	19
2	7	3	47	167
2	8	10	15	555
3	9	7	7	367
3	10	10	6	546
3	11	9	55	535
4	12	7	15	375
4	13	11	19	619
4	14	12	42	702
5	15	4	9	189
5	16	11	57	657
5	17	6	30	330
5	18	4	54	234
6	19	4	5	185
6	20	8	2	422
6	21	4	20	200
6	22	6	3	303

Figuur 15: Een voorbeeld van een databestand.

In het voorbeeld magazijn met de bijbehorende data werd geen scattered storage toegepast. Daarom was het noodzakelijk om enkele manipulaties op de verkregen data uit te voeren. Zo werd de inhoud van de kolom OpslagLocatieID beschouwd als de bestelde SKUs. Voor elke rij in elk databestand werd een willekeurig getal tussen één en vier gegenereerd die de bestelde hoeveelheid voorstelt. 'Figuur 16' werd bekomen door het voorbeeld uit 'Figuur *15*' aan te passen met bovenstaande bewerkingen.

OrderID	OrderlijnID	PickgangID	OpslagCelID	SKU	Vraag
1	1	2	32	92	2
1	2	9	46	526	1
1	3	10	36	576	4
1	4	1	13	13	1
2	5	4	50	230	3
2	6	1	19	19	2
2	7	3	47	167	2
2	8	10	15	555	2
3	9	7	7	367	2
3	10	10	6	546	1
3	11	9	55	535	4
4	12	7	15	375	4
4	13	11	19	619	4
4	14	12	42	702	1
5	15	4	9	189	3
5	16	11	57	657	2
5	17	6	30	330	1
5	18	4	54	234	2
6	19	4	5	185	3
6	20	8	2	422	1
6	21	4	20	200	2
6	22	6	3	303	2

Figuur 16: Een voorbeeld van een databestand na de aanpassingen voor een scattered storage magazijn.

Per groep van bestanden (4, 8 en 12 orderlijnen) werd vervolgens geteld hoe vaak SKUs besteld werden. Hierbij werd geen rekening gehouden met het aantal eenheden dat besteld werd per SKU. Op basis van die telling werden de volgende gegevens gevonden:

	Aantal SKUs driemaal besteld	**Aantal SKUs tweemaal besteld**	**Aantal SKUs eenmalig besteld**	**Totaal aantal SKUs**
4 orderlijnen	2	15	145	162
8 orderlijnen	1	23	194	218
12 orderlijnen	4	26	174	204

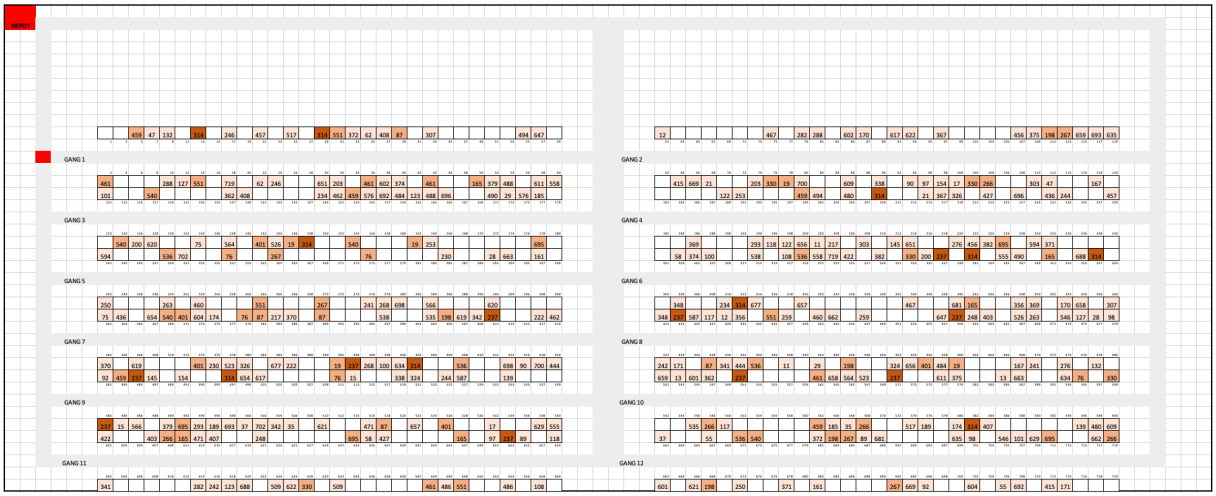
Tabel 9: Een telling van het aantal keer dat een SKU besteld werd per bestandengroep.

De SKUs werden opgedeeld in de klassen A, B en C die respectievelijk SKUs bevatten die driemaal, tweemaal en eenmalig besteld werden. Aan de hand van die drie klassen werden de opslaglocaties in het voorbeeld magazijn opgevuld met SKUs. Daarbij kregen SKUs uit klasse A, B en C respectievelijk tien, vijf en twee locaties toegewezen. De willekeurige toewijzing van een SKU aan een locatie gebeurde via een progamma geschreven in Python. Bovendien werd voor elke opslaglocatie een willekeurige voorraadscapaciteit tussen twee en vijf bepaald, zodat altijd aan de vraag van een SKU voldaan kon worden. Het voorbeeld magazijn voor de bestanden uit groep 1 (met maximaal vier orderlijnen) wordt hieronder weergegeven. De getallen in de opslaglocaties stellen de geplaatste SKU voor en de celkleur geeft de SKU-klasse weer met de volgende legende:

SKU afkomstig uit klasse A	SKU afkomstig uit klasse B	SKU afkomstig uit klasse C	Een lege opslaglocatie

Tabel 10: Een kleurlegende van de klassen in het voorbeeld magazijn.

Het ingevulde voorbeeld magazijn voor de data met maximaal vier orderlijnen:



Figuur 17: Een visuele weergave van het voorbeeld magazijn voor de bestandengroep met maximaal vier orderlijnen.

Bovenstaand voorbeeld magazijn en de ingevulde voorbeeld magazijnen van de andere datagroepen zijn in de bijlage in een groter formaat terug te vinden.

Vanaf nu zullen de bestanden worden aangehaald als picklijsten.

3.3 Testen van de prioriteitsheuristieken op nieuwe data

In dit deel zullen de prioriteitsheuristieken van Weidinger (2018) getest worden op de verkregen dataset. Dat wil zeggen dat het picker-routing probleem werd opgelost voor elk order in elk van de 30 picklijsten. Het doel hiervan is om de gevonden oplossingprestaties te vergelijken met de oplossingsprestaties gevonden in de paper van Weidinger (2018).

De eerste stap om het picker-routing probleem op te lossen is het bepalen van de te bezoeken opslaglocaties. Dat werd gedaan door de prioriteitsheuristieken uit de paper van Weidinger (2018) toe te passen. Om dit proces efficiënt en foutloos te laten verlopen, werd dit uitgevoerd door een programma geschreven in Python. De werking van het programma wordt uitgelegd a.d.h.v. een voorbeeld uit de picklijsten met maximaal vier orderlijnen. Het programma gaat van start door de afstandsmatrix in te lezen. Hierdoor kunnen tijdens de berekeningen de afstanden tussen locaties in het magazijn worden opgezocht. Vervolgens wordt de lay-out van het voorbeeld magazijn ingelezen om per locatie de opgeslagen SKU en de bijbehorende voorraad te kennen. Daarna worden elk van de tien picklijsten uit groep 1 (maximaal vier orderlijnen) ingelezen om ze samen te voegen tot een lijst van te onderzoeken orders. Voor elk van de orders in deze lijst worden de drie prioriteitsheuristieken berekend en de gevonden locaties worden geëxporteerd naar een excel-bestand. De uitkomsten van de heuristieken voor het eerste order van de eerste picklijst worden bijvoorbeeld als volgt weergegeven:

Naam order	Order 1 uit picklijst 1
SKUs (gevraagde hoeveelheid)	13 (1), 92 (2), 526 (1), 576 (4) De SKUs werden door het programma gesorteerd o.b.v. het laagste aantal opslaglocaties (en vervolgens o.b.v. index).
SinglePosition	Iteratie 1: selectie = [0, 155, 175, 467, 543, 696] Iteratie 2: selectie = [0, 155, 175, 467, 585, 696]
MinMin	Selectie = [0, 144, 155, 175, 481, 543]
MinMax	Selectie = [0, 144, 155, 175, 481, 543]

Tabel 11: Een voorbeeld van een order tijdens na het berekenen van de prioriteitsheuristieken.

Voor een uitgewerkt voorbeeld van de prioriteitsheuristieken wordt verwezen naar '2.2.3.1 Het eerste onderzoek'. Uit 'Tabel 11: ' valt af te leiden dat voor elke iteratie in de SinglePosition methode een selectie van opslaglocaties wordt bijgehouden. Daarentegen wordt voor de andere methoden, MinMin en MinMax, altijd één oplossing weggeschreven.

De tweede stap voor het oplossen van het picker-routing probleem is het vinden van een route tussen de eerder geselecteerde locaties. Hiervoor werd eveneens een programma geschreven in Python. Dat gaat net zoals het vorige programma van start door de afstandsmatrix in te lezen. Daarna leest het programma de oplossingen van de vorige stap in met de geselecteerde opslaglocaties. Voor elk order wordt de beste route berekend d.m.v. dynamic programming. Dat is een exacte oplossingsmethode die het routeringprobleem opsplitst in een reeks overlappende, stapsgewijze deelproblemen (Mahmoudi & Zhou, 2016). Voor de SinglePosition methode werden eerder meerdere selecties van opslaglocaties opgesteld. Voor elk van deze selecties wordt een route berekend, maar enkel de kortste route wordt bijgehouden als oplossing. Voor zowel de MinMin methode als de MinMax methode moet telkens maar één route berekend worden die meteen dient als de oplossing. Ten slotte worden de gevonden oplossingen opnieuw weggeschreven naar een excel-bestand om analyses op uit te voeren.

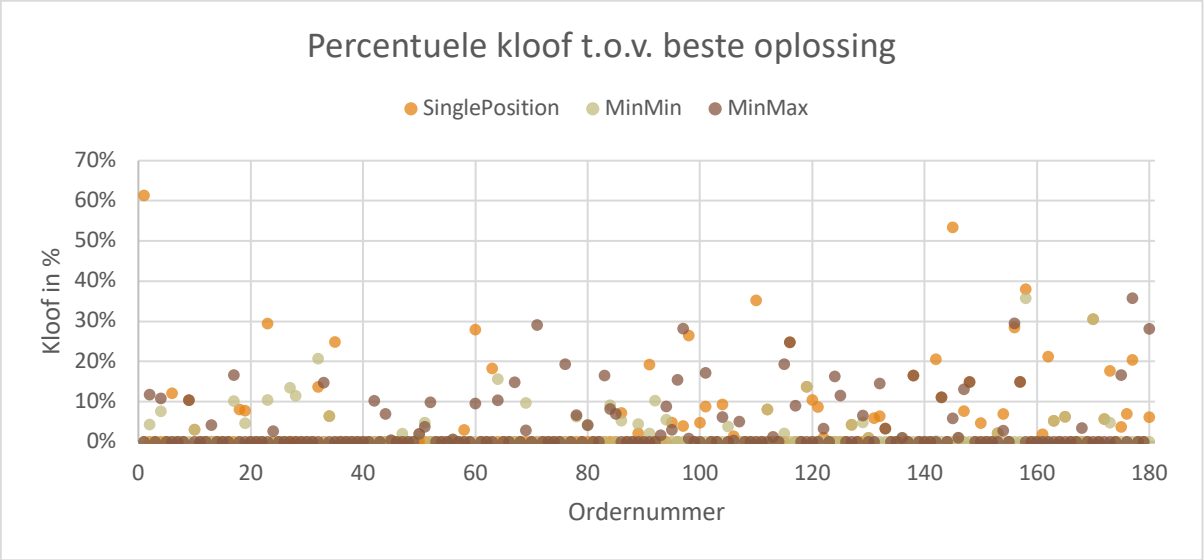
Voor elk order uit elke picklijst bestaat een optimale oplossing. Dat zijn in totaal 180 oplossingen (30 picklijsten met telkens zes orders) oftwel 60 oplossingen per bestandengroep (10 picklijsten met telkens zes orders). Toch was het niet mogelijk om deze exacte oplossingen te vinden doordat geen volledige toegang tot een oplossingsprogramma beschikbaar was. Daarom zullen de oplossingsprestaties van de drie prioriteitsheuristieken vergeleken worden door steeds de best gevonden oplossing per order te bepalen. 'Tabel 12' werd opgesteld om per heuristiek de oplossingsprestaties weer te geven. De eerste kolom '# Beste oplossing' bevat het aantal keer dat de heuristiek de best gevonden oplossing vond. Indien dat aantal vergeleken wordt met het totaal aantal oplossingen, dan wordt het percentage in de kolom '% Beste oplossing' gevonden. De derde kolom 'Gemiddelde kloof' beschrijft de gemiddelde percentuele afwijking van de gevonden oplossing t.o.v. de beste oplossing. De volgende formule werd gebruikt om per order de kloof te bereken: $percentuele\ kloof = (huidige\ oplossing - beste\ oplossing)/(beste\ oplossing)$. De laatste kolom 'Maximale kloof' bevat de grootst gevonden kloof. De oplossingsprestaties werden eveneens per picklijst-groep berekend, maar hierbij werden geen grote verschillen tussen de groepen opgemerkt. Daardoor wordt

om het overzicht te bewaren deze data enkel weergegeven in de bijlagen. Dat geldt eveneens voor alle volgende scenario's.

Heuristiek	# Beste oplossing	% Beste oplossing	Gemiddelde kloof	Maximale kloof
SinglePosition	114	63,33%	4,34%	61,35%
MinMin	139	77,22%	1,69%	35,74%
MinMax	120	66,67%	3,47%	35,76%

Tabel 12: De oplossingsprestaties voor de prioriteitsheuristieken met de nieuwe dataset.

Indien de resultaten over de verschillende heuristieken worden vergeleken, dan is het opvallend dat de MinMin heuristiek domineert in het vinden van de beste oplossing. Over alle picklijsten vindt de methode 77,22% van de beste oplossingen. Dat is minstens 10% meer dan de andere methoden. Bij SinglePosition en MinMax bedragen de percentages respectievelijk 63,33% en 66,67%. Daarnaast vindt de MinMin methode de laagste gemiddelde kloof van 1,69% en eveneens de laagste maximale kloof van 35,74%. Om te zien welke methode de grootste schommeling vertoont t.o.v. de beste oplossing werd de grafiek uit 'Figuur *18*' opgesteld. Elke bol in de grafiek stelt de gevonden percentuele kloof bij een order voor. Het valt op dat de oranje bollen van SinglePosition het hoogst lijken te liggen. Daarna volgt MinMax met minder hoge uitschieters. Ten slotte de MinMin methode vertoont slechts twee bollen die meer dan 10% boven de andere groene bollen uitstijgen.



Figuur 18: De percentuele kloof t.o.v. de beste oplossingen voor het testen van de prioriteitsheuristieken met de nieuwe dataset.

Het besluit voor het eerste scenario is als volgt. De MinMin heuristiek wordt beschouwd als de beste oplossingsmethode om de orders uit de dataset op te lossen en de SinglePosition heuristiek kan soms tot beduidend slechte resultaten leiden. Dat stemt overeen met de resultaten gevonden in de paper van Weidinger (2018). Daar werd namelijk gevonden dat bij grote datasets de MinMin heuristiek het beste presteert op vlak van zowel het aantal keren dat de beste oplossing werd gevonden als op de gemiddelde percentuele kloof. Daarentegen vond de auteur dat de MinMax methode beduidend over de grootste gemiddelde percentuele kloof beschikte, terwijl dat in de huidige bevindingen de SinglePosition heuristiek is.

Methode	**Totale route-tijd in minuten**
4 orderlijnen	528,16
8 orderlijnen	564,65
12 orderlijnen	530,54
Totaal	**1.623,36**

Tabel 13: De gevonden route-tijden in minuten bij het testen van de prioriteitsheuristieken op nieuwe data.

Indien per picklijst-groep de best gevonden route-tijden worden opgeteld, dan worden de getallen in 'Tabel 13' gevonden. Daarbij is weinig verschil in route-tijd zichtbaar tussen de verschillende picklijst-groepen ondanks dat bijvoorbeeld de picklijst-groep met maximaal twaalf orderlijnen grotere orders bevat dan de andere picklijst-groepen. Indien de gevonden tijden over de picklijst-groepen heen worden opgeteld, dan wordt een totale route-tijd van 1.623 minuten en 21,6 seconden gevonden.

3.4 Oplossingsprestaties na batching

Batching is een methode waarbij meerdere orders op een picklijst worden gegroepeerd (Gu et al., 2007). Dat kan handig zijn in een e-commerce scattered storage magazijn waar de orders typisch klein zijn (Gong & De Koster, 2008). Vooral als dezelfde SKU nodig is voor meerdere bestellingen is het vormen van batches voordelig (de Koster et al., 2007). Een order picker die zo een batch pickt, zal een efficiëntere route kunnen volgen dan de route die gevormd zou worden door diezelfde orders sequentieel te picken in meerdere routes (Weidinger et al., 2018). De batching methoden die worden getest zijn FIFO-batching en een nieuwe methode, namelijk SinglePosition batching. Deze laatste methode is gebaseerd op de SinglePosition prioriteitsheuristiek van Weidinger (2018). Om te onderzoeken of de resultaten van SinglePosition batching afhangen van het gekozen startpunt van de berekeningen worden twee verschillende methoden om het startpunt te bepalen met elkaar vergeleken. Ten slotte worden oplossingsprestaties over alle batching methoden heen met elkaar vergeleken.

3.4.1 FIFO batching

Priority rule based batching is een batching methode waarbij orders volgens hun prioriteit op de picklijst worden gezet. Bijvoorbeeld bij First-In-First-Out (FIFO) batching oftewel first-come-first-served (FCFS) batching is de prioriteitsregel als volgt: orders die eerst binnen komen, zullen eerst gepickt worden. De batchgrootte wordt vastgelegd op twee orders. Dat wil zeggen dat telkens de twee bovenste orders op de picklijst een batch zullen vormen. Deze batchgrootte werd gekozen om twee redenen. Eerst is het belangrijk dat het aantal te picken items per pickronde praktisch haalbaar is voor een order picker, want een grotere pick-kar vermindert de beweeglijkheid van de picker (Weidinger et al., 2018). Indien bijvoorbeeld uit de picklijst-groep met maximaal twaalf orderlijnen twee orders gecombineerd worden, dan kan het voorkomen dat een order picker 24 SKUs uit het magazijn moet verzamelen tijdens zijn/haar pickronde. De tweede reden voor deze batchgrootte is om het aantal geselecteerde opslaglocaties te beperken. Tussen deze opslaglocaties moet steeds een route gevormd worden met de gekozen routeringsmethode dynamic programming. Aangezien dat een exacte methode is, zou een groot aantal opslaglocaties de berekeningstijd aanzienlijk verhogen.

Het bepalen van de batches gebeurde via een Python programma. Dat programma leest alle 30 picklijsten één voor één in. In elk van die picklijsten zitten zes orders. Per picklijst worden orders simpelweg per twee samen genomen in een batch van boven naar onder in de lijst. De berekende batchnummers worden aan de picklijst toegevoegd en de vernieuwde versie van de picklijst wordt opgeslagen. Bijvoorbeeld de originele versie van de eerste picklijst uit groep 1 (maximaal vier orderlijnen) werd al eerder weergegeven in ‘Figuur 16’. De picklijst zal na het berekenen van de batches een lay-out bezitten zoals in ‘Figuur 19’. In deze figuur is een nieuwe kolom vooraan toegevoegd die het batchnummer weergeeft.

BatchID	OrderID	OrderlijnID	PickgangID	OpslagCelID	SKU	Vraag
1	1	1	2	32	92	2
1	1	2	9	46	526	1
1	1	3	10	36	576	4
1	1	4	1	13	13	1
1	2	5	4	50	230	3
1	2	6	1	19	19	2
1	2	7	3	47	167	2
1	2	8	10	15	555	2
2	3	9	7	7	367	2
2	3	10	10	6	546	1
2	3	11	9	55	535	4
2	4	12	7	15	375	4
2	4	13	11	19	619	4
2	4	14	12	42	702	1
3	5	15	4	9	189	3
3	5	16	11	57	657	2
3	5	17	6	30	330	1
3	5	18	4	54	234	2
3	6	19	4	5	185	3
3	6	20	8	2	422	1
3	6	21	4	20	200	2
3	6	22	6	3	303	2

Figuur 19: Een voorbeeld van een picklijst na FIFO-batching.

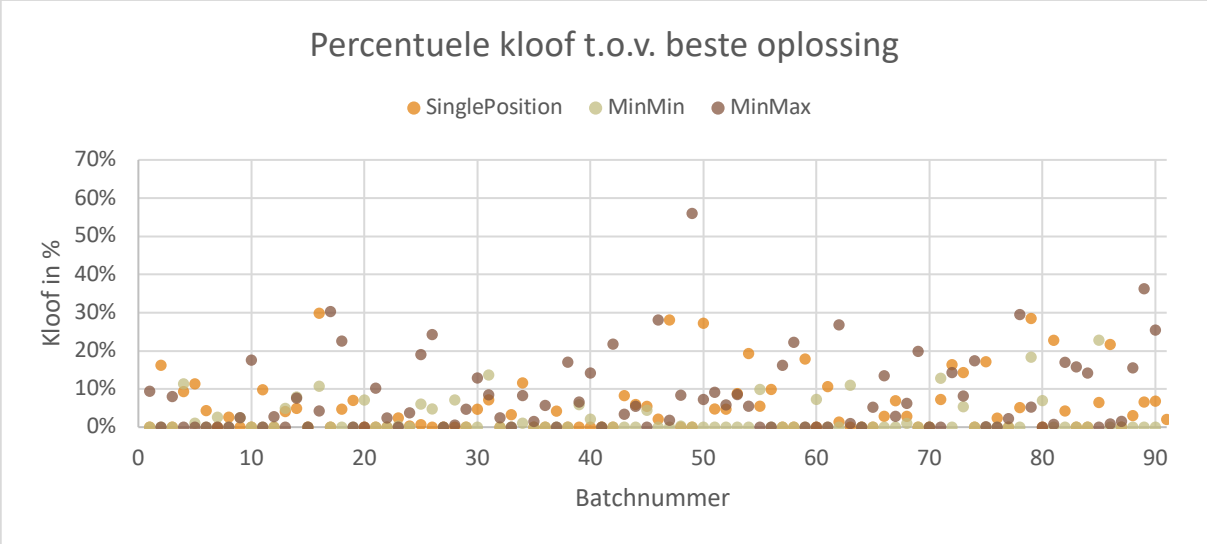
Vervolgens moet het picker-routing probleem worden opgelost voor de nieuwe data. Dat gebeurt op dezelfde manier als in ‘3.3 ’, maar nu worden de prioriteitsheuristieken berekent per batch i.p.v. per order.

Om de oplossingsprestaties van de prioriteitsheuristieken met elkaar te vergelijken wordt dezelfde overzichtstabel gebruikt als in '3.3 '. Het totale aantal beste oplossingen is nu wel verschillend. Doordat de orders per twee werden samengenomen in een batch zijn de beste oplossingen gehalveerd tot 90 in totaal (30 picklijsten met telkens drie batches).

Heuristiek	# Beste oplossing	% Beste oplossing	Gemiddelde kloof	Maximale kloof
SinglePosition	39	43,33%	5,28%	29,83%
MinMin	65	72,22%	2,10%	22,78%
MinMax	29	32,22%	8,06%	55,98%

Tabel 14: De oplossingsprestaties na FIFO-batching.

Net zoals bij het eerste scenario worden de beste resultaten behaald door de MinMin heuristiek. In totaal worden 65 van de 90 beste oplossingen gevonden of m.a.w. 72,22%. Bij SinglePosition en MinMax bedraagt dat percentage 43,33% en 32,22%. De gemiddelde kloof is opnieuw minimaal bij MinMin, die daar slechts 2,10% bedraagt, alsook de maximale kloof ter waarde van 22,78%. De hoogste maximale kloof wordt gevonden bij MinMax. Daar ligt de gevonden oplossing 55,98% boven de beste oplossing. In 'Figuur 20' is zichtbaar dat de bruine bollen van MinMax vaker hogere waarden aannemen en vandaar dat de MinMax methode de hoogste gemiddelde kloof van 8,06% vindt.



Figuur 20: De percentuele kloof t.o.v. de beste oplossingen voor de data na FIFO-batching.

Hieruit kan worden afgeleid dat de MinMin heuristiek opnieuw de beste oplossingen vindt en de MinMax methode lijkt de grootste schommelingen te vinden.

3.4.2 SinglePosition batching met de SKU met de minste opslaglocaties als startpunt

Daarnet werden bij FIFO-batching de batches willekeurig samengesteld o.b.v. hun volgorde op de picklijst. Het is daardoor mogelijk dat de gebatchte orders niet goed samen passen en dat andere combinaties van orders in batches tot een betere route-tijd zouden geleid hebben. Daarom wordt nu een methode gebruikt die hier wel rekening mee houdt, namelijk SinglePosition batching. Hierbij wordt de SinglePosition prioriteitsheuristiek uit de paper van Weidinger (2018) lichtjes aangepast om tot de nieuwe batchingstrategie te komen. Opnieuw wordt om dezelfde redenen een batchgrootte van twee orders gehanteerd.

Net zoals bij FIFO-batching werd Python gebruikt om de batches samen te stellen. De werking van het programma wordt wederom gekaderd d.m.v. de picklijst-groep met maximaal vier orderlijnen. Om te starten wordt de afstandsmatrix en de magazijn lay-out ingelezen. Vervolgens worden de tien picklijsten één voor één ingelezen. De batching gebeurt per picklijst. Bijvoorbeeld de eerste picklijst uit de picklijst-groep wordt hieronder opnieuw getoond.

OrderID	OrderlijnID	PickgangID	OpslagCelID	SKU	Vraag
1	1	2	32	92	2
1	2	9	46	526	1
1	3	10	36	576	4
1	4	1	13	13	1
2	5	4	50	230	3
2	6	1	19	19	2
2	7	3	47	167	2
2	8	10	15	555	2
3	9	7	7	367	2
3	10	10	6	546	1
3	11	9	55	535	4
4	12	7	15	375	4
4	13	11	19	619	4
4	14	12	42	702	1
5	15	4	9	189	3
5	16	11	57	657	2
5	17	6	30	330	1
5	18	4	54	234	2
6	19	4	5	185	3
6	20	8	2	422	1
6	21	4	20	200	2
6	22	6	3	303	2

Figuur 21: Een voorbeeld van een picklijst.

De batching procedure start steeds bij de bovenste nog niet gebatchte order van de picklijst. Dat is in dit geval order 1 die met de andere nog niet gebatchte orders zal vergeleken worden om een batch te kunnen vormen. De SinglePosition prioriteitsheuristiek gaat van start door de SKU te zoeken die op het minste aantal opslaglocaties beschikbaar is. Dat gebeurt voor het huidige order door de locaties van elke SKU te tellen in de magazijn lay-out. Net zoals in de paper van Weidinger (2018) wordt als tie-breaker de SKU met de laagste index gekozen. In het voorbeeld zijn de SKUs 92, 526, 576 en 13 allemaal beschikbaar op twee opslaglocaties. Bijgevolg wordt SKU 13 gekozen door de tie-breaker. Doordat deze SKU over twee opslaglocaties bezit, namelijk 543 en 585, zullen twee iteraties van de SinglePosition heuristiek worden uitgevoerd per mogelijke order om mee te batchen. In 'Tabel 15' wordt een overzicht weergegeven van de gevonden scores. De berekening voor order 2 werd volledig uitgeschreven ter verduidelijking. Tijdens elke iteratie wordt voor elke opslaglocatie van elke SKU de SinglePosition score berekend a.d.h.v. de formule $P_{SP}^{j}(i) = c_{ij} + \frac{\bar{q}-q_i}{\bar{q}}$. Voor elke SKU

wordt de laagst gevonden score aangeduid met een pijl. De scores met een pijl worden vervolgens opgeteld om daarna gedeeld te worden door het aantal SKUs. Op die manier wordt een gemiddelde score bekomen. Indien dat niet zou gebeuren, dan zouden orders met minder orderlijnen vaak een lagere score bekomen en bijgevolg meer kans hebben om eerder gebatcht te worden.

Order	Berekeningen
2	<u>Iteratie 1 met j = locatie 543:</u> SKU 230: i = locatie 285: score = 820,2 ← i = locatie 376: score = 1261,4 SKU 19: i = locatie 78: score = 1147,0 i = locatie 146: score = 1354,0 i = locatie 162: score = 1042,6 i = locatie 392: score = 949,0 ← i = locatie 460: score = 1144,4 SKU 167: i = locatie 118: score = 1927,4 i = locatie 468: score = 1300,0 ← SKU 555: i = locatie 345: score = 1405,0 i = locatie 540: score = 259,0 ← Laagste score van elke SKU opgeteld = 3328,2 Gemiddelde score = 832,05 <u>Iteratie 2 met j = locatie 585:</u> SKU 230: i = locatie 285: score = 1639,2 ← i = locatie 376: score = 2080,4 SKU 19: i = locatie 78: score = 1888,0 i = locatie 146: score = 2173,0 i = locatie 162: score = 1861,6 i = locatie 392: score = 1768,0 i = locatie 460: score = 1027,4 ← SKU 167: i = locatie 118: score = 1108,4 i = locatie 468: score = 871,0 ← SKU 555: i = locatie 345: score = 1054,0 ← i = locatie 540: score = 1078,0 Laagste score van elke SKU opgeteld = 4591,6 Gemiddelde score = 1147,9 Laagst gevonden score tijdens de iteraties = 832,05
3	Laagst gevonden score tijdens de iteraties = 898,00
4	Laagst gevonden score tijdens de iteraties = 779,86
5	Laagst gevonden score tijdens de iteraties = 741,35

6	Laagst gevonden score tijdens de iteraties = 953,40

Tabel 15: Een overzicht van de werking van SinglePosition batching.

Uit 'Tabel 15' blijkt dat order 1 samen met order 5 de eerste batch zal vormen. De volgende nog niet gebatchte order van de picklijst is order 2 die na het berekenen een batch vormt met order 3. Dan blijven ten slotte order 4 en order 6 nog over om de derde en laatste batch te vormen. Vooraan wordt een nieuwe kolom aan de picklijst toegevoegd die het batchnummer aangeeft, zie 'Figuur 22'.

BatchID	OrderID	OrderlijnID	PickgangID	OpslagCelID	SKU	Vraag
1	1	1	2	32	92	2
1	1	2	9	46	526	1
1	1	3	10	36	576	4
1	1	4	1	13	13	1
2	2	5	4	50	230	3
2	2	6	1	19	19	2
2	2	7	3	47	167	2
2	2	8	10	15	555	2
2	3	9	7	7	367	2
2	3	10	10	6	546	1
2	3	11	9	55	535	4
3	4	12	7	15	375	4
3	4	13	11	19	619	4
3	4	14	12	42	702	1
1	5	15	4	9	189	3
1	5	16	11	57	657	2
1	5	17	6	30	330	1
1	5	18	4	54	234	2
3	6	19	4	5	185	3
3	6	20	8	2	422	1
3	6	21	4	20	200	2
3	6	22	6	3	303	2

Figuur 22: Een voorbeeld van een picklijst na SinglePosition batching.

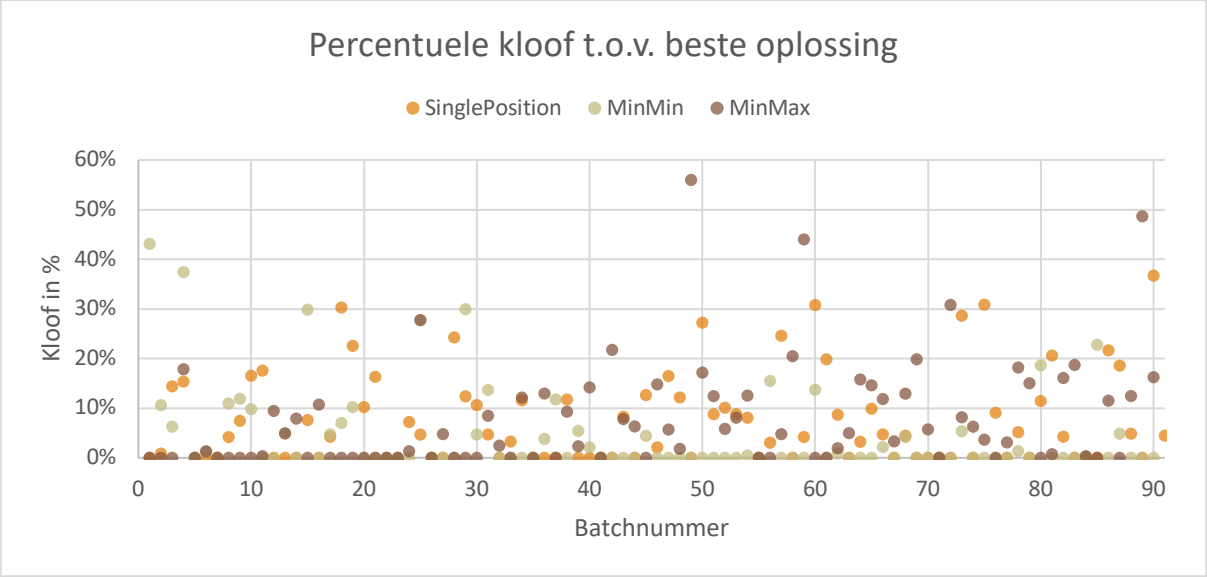
Ten slotte moet het picker-routing probleem nog worden opgelost voor de huidige picklijsten. Dat gebeurt op dezelfde manier als in '3.3 ', maar nu worden de prioriteitsheuristieken berekent per batch i.p.v. per order.

Wederom wordt dezelfde overzichtstabel gebruikt als in '3.3 '. Het aantal beste oplossingen bedraagt 90 (30 picklijsten met telkens drie batches).

Heuristiek	# Beste oplossing	% Beste oplossing	Gemiddelde kloof	Maximale kloof
SinglePosition	36	40%	7,58%	36,70%
MinMin	58	64,44%	4,24%	43,09%
MinMax	36	40%	7,65%	55,98%

Tabel 16: De oplossingsprestaties met de data na SinglePosition batching.

De beste oplossingen worden wederom het vaakst gevonden door MinMin die 58 beste oplossingen vindt oftewel 64,44%. Dat zijn 22 beste oplossingen of 24,44% meer dan SinglePosition en MinMax. De laagste kloof wordt eveneens gevonden door MinMin, maar de maximale kloof is minimaal bij SinglePosition en het hoogst bij MinMax. Op 'Figuur *23*' is te zien dat de MinMax methode vooral grote schommeling kent bij de picklijst-groepen met maximaal acht en twaalf orderlijnen (m.a.w. vanaf batchnummer 30) en slechts een lage percentuele kloof voor de andere pick-lijst groep (m.a.w. batchnummers 1 t.e.m. 30). Bij de MinMin heuristiek valt net het omgekeerde op te merken.



Figuur 23: De percentuele kloof t.o.v. de beste oplossingen voor de data na SinglePosition batching.

3.4.3 SinglePosition batching met de SKU met de meeste opslaglocaties als startpunt

De SinglePosition prioriteitsheuristiek gaat altijd van start door in het te batchen order de SKU te zoeken die op het minste aantal opslaglocaties beschikbaar is. Per opslaglocatie van deze SKU wordt een iteratie opgesteld. In deze iteraties wordt de huidige opslaglocatie gebruikt als j-waarde en de i-waarde is telkens de opslaglocatie van een andere mogelijke opslaglocatie uit een ander order met de formule $P_{SP}^{j}(i) = c_{ij} + \frac{\bar{q}-q_i}{\bar{q}}$. Alle mogelijke opslaglocaties van het andere order worden bijgevolg enkel vergeleken met de opslaglocaties van de eerst gekozen SKU uit het te batchen order. Bijgevolg heeft deze eerste keuze van SKU een invloed op de SinglePosition scores die gevonden zullen worden. Daarom wordt in dit deel de keuze van deze eerste SKU op een andere manier uitgevoerd om de invloed van de beslissing te onderzoeken. Meer specifiek zal in het te batchen order de SKU die op het meeste aantal opslaglocaties heeft gekozen worden als startpunt. Daardoor is het waarschijnlijk dat meer iteraties van de heuristiek uitgevoerd zullen moeten worden en bijgevolg meer scores gevonden worden om het order met de laagste score te kiezen.

Weidinger (2018) baseerde zijn beslissing van de eerste SKU op de mate van flexibiliteit van de verschillende SKUs. Bij een SKU met meer opslaglocaties is de kans groter dat een opslaglocatie met een lage SinglePosition score gevonden kan worden. Daarom wordt de SKU met het laagste aantal opslaglocaties gekozen als het vergelijkingspunt voor de SinglePosition heuristiek. Echter in SinglePosition batching klopt deze gedachtegang niet volledig. De SKUs waar de eerst gekozen SKU mee vergeleken wordt, komen namelijk uit een andere order waarvan de compatibiliteit om mee te batchen wordt getest. Daarom zal nu getest worden wat de invloed is om van de te batchen order de SKU met het meeste aantal opslaglocaties te kiezen als de eerste SKU. De tie-breaker blijft onveranderd, namelijk de SKU met de laagste index.

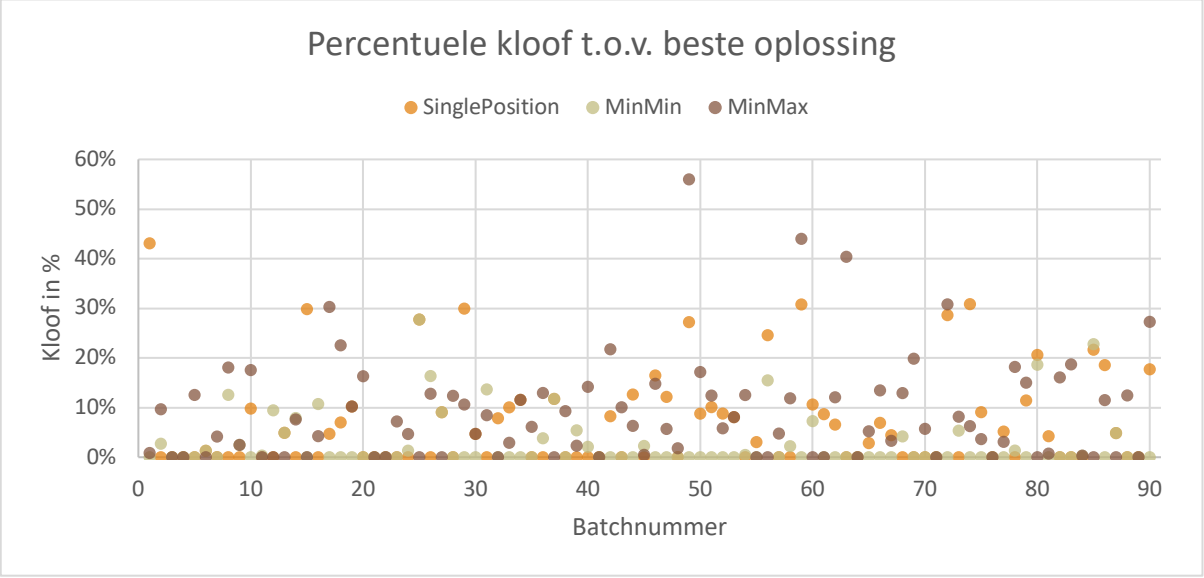
De werkwijze voor het opstellen van de batches en het oplossen van het picker-routing probleem verlopen identiek (behalve natuurlijk de keuze van de eerste SKU) als eerder uitgelegd in '3.4.2 SinglePosition batching met de SKU met de minste opslaglocaties als startpunt'.

De overzichtstabel uit '3.3 ' wordt opnieuw met de oplossingsprestaties ingevuld. Het totale aantal beste oplossingen bedraagt 90 (30 picklijsten met telkens drie batches).

Heuristiek	# Beste oplossing	% Beste oplossing	Gemiddelde kloof	Maximale kloof
SinglePosition	45	50%	6,75%	43,09%
MinMin	63	70%	2,54%	27,74%
MinMax	27	30%	8,84%	55,98%

Tabel 17: De oplossingsprestaties met de data na SinglePosition batching met de alternatieve methode voor het selecteren van de eerste SKU.

In vergelijking met de vorige SinglePosition batching methode worden bij SinglePosition en MinMin nu betere cijfers behaalt. Het aantal beste oplossingen dat door de methoden gevonden wordt, stijgt van 36 naar 45 voor SinglePosition en van 58 naar 63 voor MinMin. De MinMax methode daarentegen presteert slechter door slechts 30% van de beste oplossingen te vinden. Indien enkel naar de huidige batching methode wordt gekeken, dan behaalt MinMin opnieuw de beste cijfers door naast 70% van de beste oplossingen te vinden eveneens een gemiddelde kloof te halen van 2,54%. De slechtst presterende heuristiek is MinMax die slechts 30% van de beste oplossingen vindt en de hoogste percentuele en maximale kloof behaalt van respectievelijk 8,84% en 55,98%. 'Figuur 24' bevestigt deze cijfers door de groene bollen die het vaakst hoger liggen.



Figuur 24: De percentuele kloof t.o.v. de beste oplossingen voor de data na SinglePosition batching met een alternatieve selectiemethode voor de eerste SKU.

3.4.4 Vergelijking van de batching methoden

Om alle scenario's met elkaar te kunnen vergelijken wordt de totaal afgelegde tijd in minuten berekend per picklijst-groep door telkens de best gevonden route-tijden op te tellen. De uitkomsten zijn zichtbaar in 'Tabel 18'.

Methode	4 orderlijnen	8 orderlijnen	12 orderlijnen	Totaal
FIFO batching	358,03	383,04	374,86	**1.115,93**
SinglePosition batching – minste locaties	346,19	383,60	368,51	**1.098,30**
SinglePosition batching – meeste locaties	342,94	381,45	363,31	**1.087,7**

Tabel 18: De totaal afgelegde route-tijd in minuten voor elk scenario.

Bij het eerste scenario werd een totaal afgelegde route-tijd van 1.623 minuten en 22 seconden gevonden. Deze totale tijd wordt bij alle batching methoden verbetert en eveneens per picklijst-groep wordt altijd een verbetering van de afgelegde afstand waargenomen. De kleinste verbetering wordt waargenomen bij FIFO-batching met een totale route-tijd van 1.098 minuten en 18 seconden. De SinglePosition batching methoden presteren beiden beter dan FIFO-batching met een route-tijd van 1.098 minuten en 18 seconden indien het startpunt de SKU is die op het minste aantal opslaglocaties beschikbaar is en een route-tijd van 1.087 minuten en 42 seconden indien het startpunt de SKU is die op het meeste aantal opslaglocaties beschikbaar is. Bijgevolg wordt de laagste totale route-tijd gevonden bij de tweede formulering van singleposition batching. Deze methode vindt eveneens de laagste route-tijden per picklijst-groep in vergelijking met alle batching methoden. Al valt het op te merken dat de andere SinglePosition batching methode een totale route-tijd vindt die slechts 1% hoger ligt.

Toch lijkt het dat de selectie van de eerste SKU weldegelijk een invloed heeft op de oplossingprestaties. Voor SinglePosition batching lijkt het daarom voordeliger om als startpunt een SKU te selecteren die meer opslaglocaties bevat. Voor elk van deze opslaglocaties wordt dan een iteratie voltooid die een score berekend. Bijgevolg hoe meer opslaglocaties, hoe meer iteraties en hoe meer scores om de laagste score van te kiezen.

Naast SinglePosition batching zouden eveneens MinMin batching en MinMax batching mogelijk zijn. De gedachtegang achter deze laatste twee methoden is verschillend van die van SinglePosition batching. Bij SinglePosition batching wordt vertrokken vanuit de selectie van één van de SKUs uit de te batchen order en de opslaglocaties van deze SKU worden vergeleken met alle mogelijke bezoekbare opslaglocaties van de andere nog niet gebatchte orders. Bij MinMin en MinMax batching zouden eerst de respectievelijke heuristieken moeten worden uitgevoerd op de te batchen order zodat een verzameling van de te bezoeken opslaglocaties bekend is. Dan zou vertrekkend van deze selectie de MinMin en MinMax prioriteitsheuristiek worden toegepast op de te batchen orders om een score per order te bekomen. Het is bijgevolg complexer om MinMin of MinMax batching toe te passen,

omdat de heuristiek eveneens gebruikt moet worden om het startpunt te bepalen. Al is het niet onmogelijk en worden deze methoden daarom als verder onderzoek aangeraden.

3.5 Oplossingsprestaties na een gelijktijdige beslissinging van opslaglocaties

Het selecteren van opslaglocaties in het picker-routing probleem gebeurt door de prioriteitsheuristieken op een sequentiële basis. Dat wil zeggen dat de selectie van de opslaglocaties wordt beïnvloedt door de planningsvolgorde van de SKUs. Bij de methoden van Weidinger (2018) wordt de SKU met het laagste aantal opslaglocaties gekozen als de eerste SKU om de opslaglocaties van te bepalen. In dit deel zal onderzocht worden wat de invloed is van deze beslissing gelijktijdig te nemen voor alle SKUs. Twee methoden zullen besproken worden die verschillen op vlak van het aantal paden dat tussen opslaglocaties wordt geteld. De formuleringen worden getest via het oplossingsprogramma Lingo en beschikken over eenzelfde notatie die zichtbaar is in 'Tabel 19'.

Notatie	Uitleg
$\boldsymbol{M = \{SKU_1, SKU_2, \ldots, SKU_k\}}$	De verzameling van de te picken SKUs (index k).
$\boldsymbol{N_k}$	De verzameling van opslaglocaties met een item van SKU_k.
$\boldsymbol{N}$	De verzameling van alle opslaglocaties van alle te picken SKUs en het depot (index 0). Deze verzameling is gesorteerd o.b.v. index.
$\boldsymbol{c_{ij}}$	De tijd om van opslaglocatie i naar opslaglocatie j te verplaatsen.
$\boldsymbol{q_i}$	De voorraad in opslaglocatie i.
$\boldsymbol{r_k}$	De te picken hoeveelheid van SKU_k.
$\boldsymbol{X_i}$	Een integere variabele die gelijk is aan het aantal items die van opslaglocatie i gepickt wordt.
$\boldsymbol{Y_{ij}}$	Een integere variabele die aangeeft indien opslaglocatie i en opslaglocatie j opeenvolgend worden bezocht. Deze variabele neemt de waarde 2 aan indien een order picker van opslaglocatie i naar opslaglocatie j gaat en dan terugkeert naar opslaglocatie i (of omgekeerd). Indien de variabele gelijk is aan 1, dan gaat de order picker van opslaglocatie i naar opslaglocatie j (of omgekeerd). Ten slotte, indien de variabele gelijk is aan nul, dan worden de opslaglocaties niet opeenvolgend bezocht.
$\boldsymbol{Z_i = \{0, 1\}}$	Een binaire variabele die aangeeft indien opslaglocatie i geselecteerd is of m.a.w. bezocht wordt.

Tabel 19: De notatie bij de wiskundige formuleringen van de gelijktijdige opslaglocatie-selectie.

3.5.1 Algemene onderschatting

De eerste formulering die werd opgesteld om de selectie gelijktijdig te laten verlopen staat hieronder in wiskundige symbolen. De formulering gaat van start door de doelfunctie te definiëren. Deze doelfunctie tracht de afgelegde afstand in het magazijn te minimaliseren. De doelfunctie wordt opgesteld door alle mogelijke paden tussen de opslaglocaties van de SKUs op de picklijst en het depot te vermenigvuldigen met de afstand tussen de locaties (1). Uiteindelijk zullen alleen maar de geselecteerde paden uit deze doelfunctie worden opgeteld doordat de andere paden een nul-waarde krijgen en de afstand bijgevolg wegvalt. Daarna start de oplijsting van beperkingen. Beperkingen (2) en (3) zorgen voor de selectie van opslaglocaties. Dat gebeurt door na te gaan dat de gevraagde hoeveelheid van elke SKU steeds gepickt wordt (2) terwijl de voorraadsbeperking van elke opslaglocatie niet overschreden wordt (3). Beperking (4) en (5) trachten een route te vormen tussen de geselecteerde opslaglocaties. Eerst en vooral moet de route starten en eindigen in het depot (4). Daarna volgen de beperkingen die voor elke geselecteerde opslaglocatie een in- en uitgaand pad selecteren (5). Ten slotte worden de variabelen die de opslaglocatie-selectie aangeven als binaire variabelen gedefinieerd (6). In de bijlagen is een ingevuld voorbeeld van de wiskundige formulering in Lingo te vinden.

(1) $$Min\ Z(Y) = \sum_{i \in N} \sum_{j=i+1}^{N} c_{ij} * Y_{ij}$$

Onderworpen aan:

(2) $$\sum_{i \in N_k} X_i = r_k \qquad \forall k \in M$$

(3) $$X_i \leq q_i * Z_i \qquad \forall k \in M, \forall i \in N_k$$

(4) $$\sum_{i \in N \setminus \{0\}} Y_{0i} = 2$$

(5) $$\sum_{j \in N: i<j} Y_{ij} + \sum_{j \in N: i>j} Y_{ji} = 2 * Z_i \qquad \forall k \in M, \forall i \in N_k$$

(6) $$X_i \geq 0 \qquad \forall i \in N \setminus \{0\}$$

(7) $$Y_{ij} \geq 0 \qquad \forall i \in N \setminus \{0\}, \forall j \in N \setminus \{0\}$$

(8) $$Z_i \in \{0, 1\} \qquad \forall i \in N \setminus \{0\}$$

Het nadeel van deze formulering is dat niet altijd één route tussen de geselecteerde opslaglocaties startend en eindigend in het depot wordt gevormd, maar wel meerdere subtours tussen de geselecteerde opslaglocaties. Doordat deze subtours zich kunnen vormen wordt de afgelegde afstand onderschat. In principe kunnen nadat deze subtours zijn vastgesteld nog beperkingen worden opgesteld die ervoor zorgen dat de subtours tussen opslaglocaties onmogelijk worden. De subtour in 'Figuur 25' kan bijvoorbeeld vermeden worden door de beperking $Y_{AB} < 2$ toe te voegen. Deze beperkingen werden echter niet toegevoegd, omdat het probleem complexer wordt om op te lossen.

Indien bijvoorbeeld de huidige subtours worden vermeden door beperkingen toe te voegen, dan kunnen opnieuw andere subtours zich vormen die opnieuw door beperkingen vermeden moeten worden. De berekeningstijd van het probleem kan daardoor enorm oplopen. Een andere mogelijkheid is om de beperkingen al toe te voegen voordat bekend is welke subtours zich voordoen in de oplossing. Een beperking moet dan worden toegevoegd voor elke mogelijke subtour en dat zal teweeg brengen dat de formulering enorm uitbreid in lengte en complexiteit. De berekeningstijd zal dan eveneens toenemen. De subtours van grootte twee, zoals het voorbeeld in 'Figuur 25: Een voorbeeld van subtours bij de onderschattende formulering.', zouden wel makkelijk voorkomen worden door beperking (7) te veranderen naar $Y_{ij} \in \{0,1\}\ \forall i \in N\backslash\{0\}, \forall j \in N\backslash\{0\}$. De Y_{ij} variabelen zijn dan binair voor alle opslaglocaties, behalve het depot. Deze variabele mag voor het depot niet binair zijn, omdat de route depot-opslaglocatie-depot wel nog moet kunnen voorkomen.

Figuur 25: Een voorbeeld van subtours bij de onderschattende formulering.

Na de selectie van de opslaglocaties door de formulering moet nog een route tussen deze locaties worden bepaald. Hiervoor wordt dynamic programming gebruikt in Python zoals uitgelegd in '3.3 '.

3.5.2 Algemene overschatting

De tweede formulering is zeer gelijkend aan de eerste formulering.

(1) $$Min\ Z(Y) = \sum_{i \in N} \sum_{j=i+1}^{N} c_{ij} * Y_{ij}$$

Onderworpen aan:

(2) $$\sum_{i \in N_k} X_i = r_k \qquad \forall k \in M$$

(3) $$X_i \leq q_i * Y_i \qquad \forall k \in M, \forall i \in N_k$$

(4) $$Y_i + Y_j \leq 1 + Y_{ij} \qquad \forall i \in N\backslash\{0\}, \forall j \in N\backslash\{0\}, i < j$$

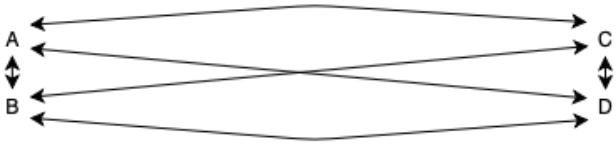
(5) $$X_i \geq 0 \qquad \forall i \in N\backslash\{0\}$$

(6) $$Y_{ij} \geq 0 \qquad i \in N\backslash\{0\}, \forall j \in N\backslash\{0\}$$

(7) $$Z_i \in \{0,1\} \qquad \forall i \in N\backslash\{0\}$$

Het doel is nu niet langer om een route te vormen. Bijgevolg is de beperking over het depot als het start- en eindpunt van de route weggelaten. Het huidige doel is om in beperking (4) alle paden tussen geselecteerde opslaglocaties te selecteren, zodat deze worden opgeteld in de doelfunctie. Het nadeel van deze methode is dat te veel paden tussen locaties worden geteld. Indien bijvoorbeeld

locatie A, B, C en D geselecteerd zijn, dan geven de pijlen alle paden aan die geteld worden in de doelfunctie. Met andere woorden wordt de afgelegde afstand nu overschat.



Figuur 26: Een voorbeeld van de geselecteerde paden bij de overschattende methode.

De routebepaling tussen de geselecteerde locaties verloopt identiek als bij de vorige formulering.

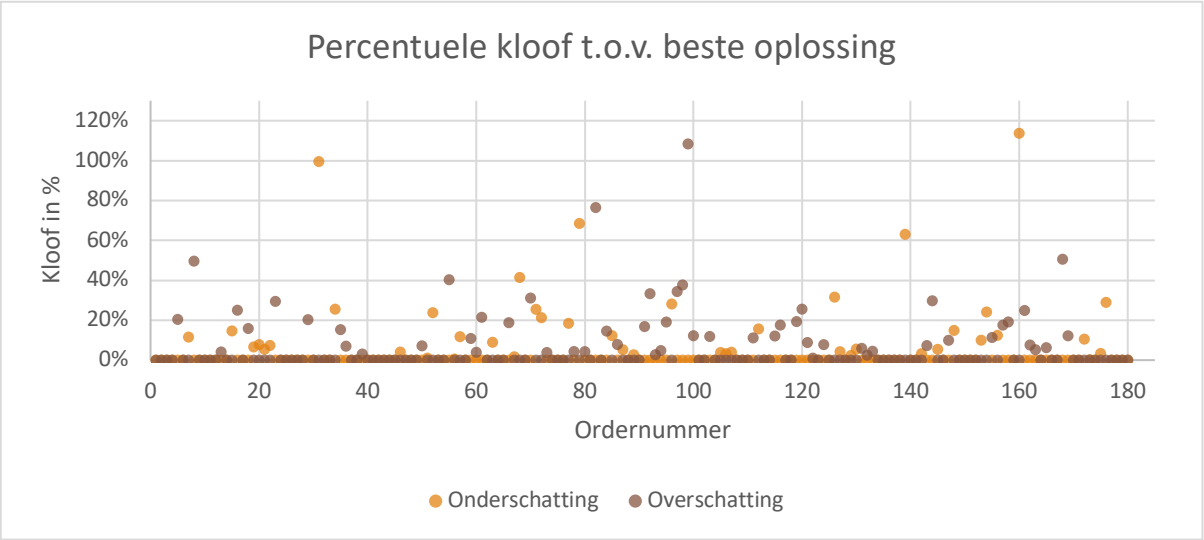
3.5.3 Vergelijking van de parallelle selectie methoden

De overzichtstabel uit '3.3 ' worden opnieuw gebruikt. De beste oplossing wordt in dit geval geselecteerd door de laagste gevonden oplossing tussen de onder- en overschattende methode te kiezen. Het totale aantal beste oplossingen bedraagt opnieuw 180 doordat geen batching op de picklijsten heeft plaatsgevonden (30 picklijsten met telkens zes orders).

Heuristiek	# Beste oplossing	% Beste oplossing	Gemiddelde kloof	Maximale kloof
Onderschatting	137	76,11%	4,52%	114%
Overschatting	100	55,56%	5,74%	108,34%

Tabel 20: De oplossingsprestaties voor de parallelle opslaglocatie-selectie.

De onderschattende methode vindt 137 keer de beste oplossing, dat komt neer op 76,11%. Bij de overschattende methode ligt dat percentage lager, namelijk 55,56%. Het is eveneens de onderschattende methode die de laagste gemiddelde kloof vindt van 4,52%. Al is het slechts een klein verschil met de overschattende methode die een gemiddelde kloof van 5,74% behaalt. De grootste kloof wordt dan wel weer gevonden bij de overschattende methode. Op 'Figuur 27' is zichtbaar dat de methoden vergelijkbare kloven vinden en dat beide methoden soms oplossingen vinden die ver van de beste oplossing af liggen.



Figuur 27: De percentuele kloof t.o.v. de beste oplossingen voor de data na een parallelle opslagselectie.

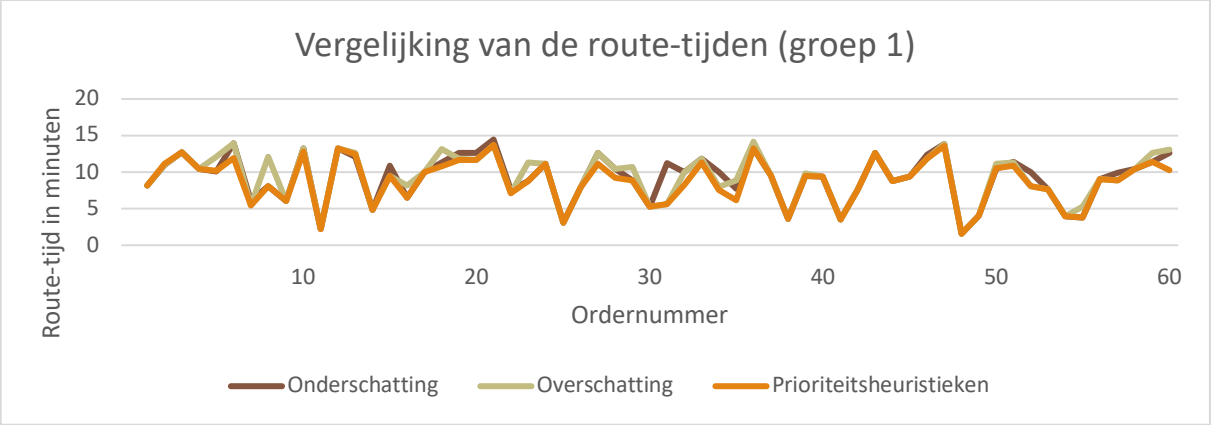
Om de onder- en overschattende methode met elkaar te kunnen vergelijken wordt opnieuw de totaal afgelegde tijd in minuten berekend per picklijst-groep. De uitkomsten zijn zichtbaar in 'Figuur *21*'.

Methode	4 orderlijnen	8 orderlijnen	12 orderlijnen	Totaal
Algemene onderschatting	556,66	583,92	568,08	**1.708,66**
Algemene overschatting	561,18	602,38	568,03	**1.731,59**

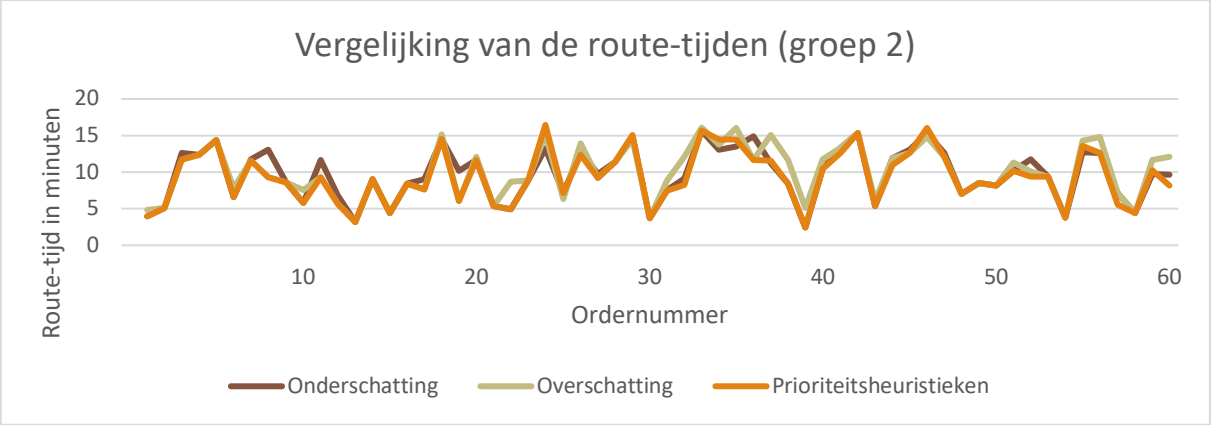
Tabel 21: Een overzicht van de afgelegde tijden in minuten per methode en per picklijst-groep.

De methoden waarbij de opslaglocatie-selectie parallel gebeurt presteren verrassingwekkend slechter dan het eerste scenario die een route-tijd van 1.623 minuten en 21,6 seconden vond. Zo wordt bij de onderschattende methode een route-tijd van 1.708 minuten en 40 seconden gevonden en bij de overschattende methode een route-tijd van 1.731 minuten en 35 seconden. Dat betekent dat de route-tijd stijgt met 5,25% voor de algemene onderschatting methode en met 6,67% voor de algemene overschatting methode. Hieruit kan besloten worden dat het voor de huidige dataset beter is om de opslaglocatie-selectie beslissing sequentieel te nemen met de prioriteitsheuristieken van Weidinger (2018).

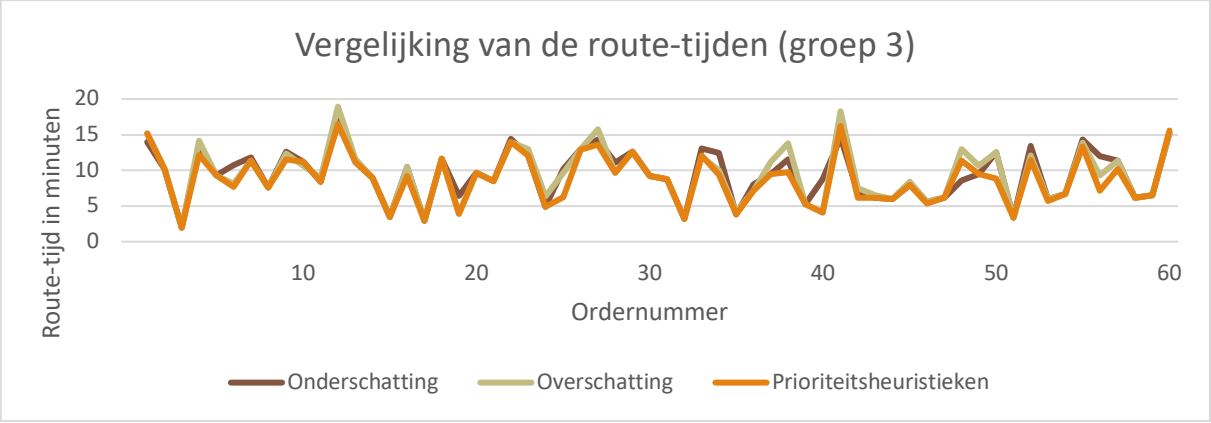
Om na te gaan of uitschieters in de route-tijden (m.a.w. route-tijden die ver af liggen van de best gevonden route-tijd) de resultaten van de onder- en overschattende methode negatief beïnvloeden worden de methoden vergeleken met het eerste scenario. Dat gebeurt door de route-tijden per order weer te geven in 'Figuur 28', 'Figuur 29' en 'Figuur 30'.



Figuur 28: Een vergelijking van de route-tijden gevonden bij de picklijsten met maximaal vier orderijnen voor scenario 1, 5 en 6.



Figuur 29: Een vergelijking van de route-tijden gevonden bij de picklijsten met maximaal acht orderijnen voor scenario 1, 5 en 6.



Figuur 30: Een vergelijking van de route-tijden gevonden bij de picklijsten met maximaal twaalf orderijnen voor scenario 1, 5 en 6.

In bovenstaande grafieken valt op te merken dat de bruine en groene lijn regelmatig net boven de oranje lijn ligt. Daar presteren de onder- en/of overschattende methode slechter dan de prioriteitsheuristieken uit het eerste scenario. Echte uitschieters t.o.v. de oranje lijn zijn niet zichtbaar, dat betekent dat de resultaten van de onder- en overschattende methode hierdoor niet negatief beïnvloed zijn.

4 Conclusie

E-commerce wint op zowel Belgisch, Europees als wereldwijd vlak aan belang (Comeos, 2020; Cramer-Flood & eMarketer, 2020; Eurostat, 2020). Door het gemak van e-commerce is een trend ontstaan om steeds frequenter, maar in kleinere hoeveelheden te bestellen (Gong & De Koster, 2008). Deze stijging in het aantal bestellingen legt extra druk op het leveringssysteem (Holland, 2019). Daarom zullen magazijnen genoodzaakt zijn om operationele kosten te minimaliseren. Scattered storage is een voorbeeld van een innovatieve toepassing om het order picking proces, dat gewoonlijk de grootste operationele kost is in een magazijn, te optimaliseren (Gu et al., 2007; Weidinger, 2018). Eenzelfde SKU wordt bij deze opslagmethode als afzonderlijke artikels over de reklocaties verspreid en bevinden zich daardoor op meerdere plaatsen in het magazijn. Dat is in tegenstelling tot traditionele opslagmethoden, zoals dedicated storage, waarbij elk product slechts in een beperkt gebied van het magazijn beschikbaar is en eventueel zelfs een vaste locatie toegewezen krijgt. Kleine orders worden significant sneller gepickt en de klant ontvangt bijgevolg sneller de gemaakte bestelling (Weidinger & Boysen, 2018). Nog maar weinig onderzoek werd uitgevoerd naar scattered storage en daarom werd volgende onderzoeksvraag opgesteld: *Hoe kan in een scattered storage picker-to-parts magazijn van een business-to-consumer e-commerce bedrijf het pickproces optimaal georganiseerd worden?*

In het eerste deel van de literatuurstudie werd een antwoord gezocht op de eerste deelvraag, namelijk 'Welke factoren beïnvloeden het pickproces?'. De factoren zijn op te delen in twee delen. Het eerste deel betrekt de factoren over de lay-out van het magazijn: de structuur, de afmetingen, de departementen en ten slotte de apparatuur. Het tweede deel zijn factoren i.v.m. de operationele werking van het magazijn: beslissingen omtrent stockering, de toepassing van wave, batch en/of zone picking en ten slotte de gekozen methode van routering (Gu et al., 2007).

Het tweede deel van de literatuurstudie beschrijft het huidige onderzoek naar scattered storage dat op te delen is in twee secties. De eerste sectie gaat over de paper van Weidinger & Boysen (2018) die het stockeringsprobleem onderzocht. Dat probleem tracht de SKUs in het magazijn optimaal te spreiden. Een heuristieke oplossingsmethode werd voorgesteld die de maximale afstand af te leggen tot een item van een SKU vanuit verschillende punten minimaliseert. De heuristiek blijkt haalbaar om toe te passen voor kleine datasets, maar voor grote datasets wordt het probleem best opgedeeld in sub-problemen. Indien de stockering via de heuristiek wordt vergeleken met een volledig willekeurige stockeringsmethode, dan wordt gevonden dat via de heuristieke stockering over het algemeen kortere pickroutes worden gevonden. De tweede sectie ontleed twee studies rond het opslaglocatie-selectie gedeelte van het picker-routing probleem. De eerste studie werd uitgevoerd door Weidinger (2018) die drie prioriteitsheuristieken voorstelde, namelijk SinglePosition, MinMin en MinMax. Bij een kleine dataset presteerde MinMax het beste en bij een grote dataset was dat de MinMin heuristiek. De tweede studie door Weidinger et al. (2018) bouwt verder op de eerste studie en stelde twee oplossingsmethoden voor het opslaglocatie-selectie probleem voor. De eerste methode was een nearest-neighbour search en de tweede een pool-based constructie heuristiek. Van deze methoden behaalde de pool-based constructie heuristiek de beste oplossingen. Bij grotere

datasets kon de berekeningstijd wel oplopen, indien een minimale berekeningstijd gewenst is, dan wordt de nearest-neighbour search aangeraden.

Het derde en laatste deel van de literatuurstudie had als doel om op zoek te gaan naar toepassingen die gelijkaardig zijn aan het picker-routing probleem in een scattered storage magazijn. Hierbij werden twee toepassingen besproken die beide een groep te bezoeken klanten opdelen in clusters. Zo een cluster is vergelijkbaar met alle opslaglocaties die een specifieke SKU bevatten. Zo kan een picklijst voltooid worden door voor elke te picken SKU de bijbehorende cluster van opslaglocaties te bezoeken. De eerste toepassing is het General Traveling Salesman Problem (GTSP). Hierbij worden de klanten opgedeelt in clusters en het voertuig moet elke cluster precies één keer bezoeken (Karapetyan & Gutin, 2012). De tweede toepassing is het General Vehicle Routing Problem (GVRP) waarbij het doel is om de optimale route te vinden voor elk van de voertuigen uit het wagenpark met vertrek- en eindpunt in het depot, die precies één klant uit elke groep bezoekt en dat rekening houdend met de capaciteitsbeperkingen van de wagens (Bektaş et al., 2011; Hà et al., 2014; P. C. Pop et al., 2013). Het GVRP reduceert zich tot het GTSP wanneer het wagenpark slechts uit één wagen met onbeperkte capaciteit bestaat (Pintea et al., 2011).

Tijdens de praktijkstudie werden drie doelen onderzocht die elk gebaseerd zijn op het picker-routing probleem in een scattered storage magazijn. Hiervoor werden in totaal zes scenario's uitgevoerd met de verkregen realistische dataset.

Het eerste doel was om de prioriteitsheuristieken uit de paper van Weidinger (2018), die de opslaglocatie-selectie uit het picker-routing probleem oplossen, te testen op een nieuwe dataset. Dat gebeurde in het eerste scenario. Uit de resulaten bleek dat de MinMin heuristiek de beste oplossingsmethode is en dat de SinglePosition heuristiek soms tot beduidend slechtere oplossingen leidt. Deze resultaten stemmen overeen met die uit de paper van Weidinger (2018), behalve dat daar de MinMax methode het vaakst tot slechtere oplossingen leidde. Indien de route-tijden over alle picklijsten werden opgeteld, dan werd een totale tijd van 1.623 minuten en 22 seconden gevonden.

Als tweede doel werd een analyse uitgevoerd van het effect van batching op de resultaten van het picker-routing probleem. Een batchgrootte van twee orders werd gehanteerd, zodat de order pickers niet overbelast kunnen worden en het probleem niet te complex werd om op te lossen. Indien FIFO-batching werd toegepast, dan werd de route-tijd van het eerste scenario van 1.623 minuten en 22 seconden verlaagd naar 1.098 minuten en 18 seconden. Daarna werd een nieuwe batching methode gebaseerd op de SinglePosition prioriteitsheuristiek voorgesteld. De SinglePositon batching methode werd getest met twee verschillende startpunten. Indien als startpunt de SKU met het minste aantal opslaglocaties werd gekozen, dan werd een route-tijd van 1.098 minuten en 18 seconden gevonden en indien het startpunt de SKU was met het meeste aantal opslaglocaties, dan werd een route-tijd van 1.087 minuten en 42 seconden gevonden. Bijgevolg is SinglePosition batching met als startpunt de SKU met het meeste aantal opslaglocaties de best gevonden batching methode voor de huidige dataset, maar het verschil met SinglePosition batching met het andere startpunt is slechts klein.

Het derde en laatste doel van de praktijkstudie was om de opslaglocatie-selectie in het picker-routing probleem op te lossen d.m.v. een parallelle/gelijktijdige beslissing voor alle te picken SKUs. Dat is in tegenstelling tot de sequentiële beslissing van de prioriteitsheuristieken van Weidinger (2018) waarbij de opslaglocatie-selectie SKU per SKU gebeurt. Twee formuleringen werden opgesteld. De eerste formulering trachtte een route te maken tussen de geselecteerde opslaglocaties, maar had als nadeel dat subtours zich konden vormen en daardoor werd de afgelegde afstand onderschat. De tweede formulering daarentegen probeert geen route te maken, maar telde alle mogelijke paden op tussen de geslecteerde opslaglocaties. Daardoor werd de route-tijd daar overschat. Na de opslaglocatie-selectie via de formuleringen werd een route gevormd tussen de locaties en de resultaten van de formuleringen zijn als volgt. De onderschattende methode vindt een route-tijd van 1.708 minuten en 40 seconden en de overschattende methode vindt een route-tijd van 1.731 minuten en 35 seconden. Het verschil tussen de methoden is bijgevolg klein, maar ze presteren beide duidelijk slechter dan de route-tijd van 1.623 minuten en 22 seconden uit het eerste scenario. Zo vond de onderschattende methode een route-tijd die 5,25% hoger lag dan het eerste scenario en de overschattende methode vond een stijging van 6,67%. Verrasingwekkend lijkt de conclusie hier dat voor de huidige dataset de gelijktijdige beslissing geen verbetering met zich meebrengt in functie van de route-tijd.

Door de veelbelovende resultaten bij batching wordt aanbevolen om dat onderzoek nog verder uit te breiden. Dat kan door bijvoorbeeld door meer batching methoden te testen, zoals MinMin batching en MinMax batching. Met nadruk op de MinMin prioriteitsheuristiek die goede resulaten vertoonde bij het opslaglocatie-selectie probleem. Daarnaast kunnen de batching methoden op een grotere dataset getest worden. Eveneens kunnen de scenario’s met de parallelle opslaglocatie-selectie grondiger getest worden. Meer specifiek de onderschattende methode, die beter presteerde als de overschattende methode, had als nadeel dat subtours gevormd werden. Indien deze subtours worden vermeden met additionele beperkingen, dan zou het mogelijk zijn dat de oplossingsprestaties van deze fomulering verbeteren.

5 Bibliografie

Aldrich, M. (2011). Online Shopping in the 1980s. *US IEEE 'Annals of the History of Computing.'*, *33*, 57–61.

Baldacci, R., Bartolini, E., & Laporte, G. (2010). Some applications of the generalized vehicle routing problem. *The Journal of the Operational Research Society*, *61*(7), 1072–1077. http://dx.doi.org/10.1057/jors.2009.51

Baniasadi, P., Foumani, M., Smith-Miles, K., & Ejov, V. (2020). A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *European Journal of Operational Research*, *285*(2), 444–457. https://doi.org/10.1016/j.ejor.2020.01.053

Bayles, D. L., & Bhatia, H. (2000). *E-Commerce Logistics & Fulfillment: Delivering the Goods*. Prentice Hall PTR.

Bektaş, T., Erdoğan, G., & Røpke, S. (2011). Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem. *Transportation Science*, *45*(3), 299–316. https://doi.org/10.1287/trsc.1100.0352

Belgreen. (z.d.). Geraadpleegd 16 maart 2021, van https://sellfy.com/belgreen/

Cambridge: Cambridge University Press. (1995). *Cambridge Dictionary*. https://dictionary.cambridge.org/

Cardona, L. F., & Gue, K. R. (2020). Layouts of Unit-Load Warehouses with Multiple Slot Heights. *Transportation Science*, *54*(5), 1332–1350. https://doi.org/10.1287/trsc.2020.0993

Comeos. (2020). *E-commerce survey 2020*. https://www.comeos.be/research/376488/E-commercestudie-2020

Cramer-Flood, E. & eMarketer. (2020). *Global Ecommerce 2020*. https://www.emarketer.com/content/global-ecommerce-2020

De Kelver, A. (2006). *Warehouses: Getuigen van welvaart*. Uitgeverij Lannoo nv.

de Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, *182*(2), 481–501. https://doi.org/10.1016/j.ejor.2006.07.009

Engelbregt, J., & Kruijer, N. (2009). *Warehousing en fysieke distributie* (Vol. 1–4). Boom onderwijs.

Eurostat. (2020). *E-commerce statistics for individuals*. https://ec.europa.eu/eurostat/statistics-explained/pdfscache/14386.pdf

Ghiani, G., & Improta, G. (2000). An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, *122*(1), 11–17. https://doi.org/10.1016/S0377-2217(99)00073-9

Gong, Y., & De Koster, R. (2008). A polling-based dynamic order picking system for online retailers. *IIE Transactions*, *40*(11), 1070–1082. https://doi.org/10.1080/07408170802167670

Gu, J., Goetschalckx, M., & McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, *177*(1), 1–21. https://doi.org/10.1016/j.ejor.2006.02.025

Gu, J., Goetschalckx, M., & McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, *203*(3), 539–549. https://doi.org/10.1016/j.ejor.2009.07.031

Guo, X., Yu, Y., & De Koster, R. B. M. (2016). Impact of required storage space on storage policy performance in a unit-load warehouse. *International Journal of Production Research*, *54*(8), 2405–2418. https://doi.org/10.1080/00207543.2015.1083624

Hà, M. H., Bostel, N., Langevin, A., & Rousseau, L.-M. (2014). An exact algorithm and a metaheuristic for the generalized vehicle routing problem with flexible fleet size. *Computers & Operations Research*, *43*, 9–19. https://doi.org/10.1016/j.cor.2013.08.017

Holland, C. (2019). The changing world of warehousing, distribution and fulfillment. *Lehigh Valley Business*. https://search.proquest.com/docview/2246578004?accountid=27889

Jane, C.-C., & Laih, Y.-W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. *European Journal of Operational Research*, *166*(2), 489–496. https://doi.org/10.1016/j.ejor.2004.01.042

Kara, I., & Bektaş, T. (2003). *Integer Linear Programming Formulation of the Generalized Vehicle Routing Problem*.

Karapetyan, D., & Gutin, G. (2012). Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem. *European Journal of Operational Research*, *219*(2), 234–251. https://doi.org/10.1016/j.ejor.2012.01.011

Kim, D. J., Ferrin, D. L., & Rao, H. R. (2009). Trust and Satisfaction, Two Stepping Stones for Successful E-Commerce Relationships: A Longitudinal Exploration. *Information Systems Research; Linthicum*, *20*(2), 237–257. http://search.proquest.com/docview/208159468/abstract/ECAD827286E2466DPQ/1

Kumar, S., & Petersen, P. (2006). Impact of e-commerce in lowering operational costs and raising customer satisfaction. *Journal of Manufacturing Technology Management*, *17*(3), 283–302. https://doi.org/10.1108/17410380610648263

Lesboek heftruck/reachtruck. (2016). https://docplayer.nl/67827554-Lesboek-heftruck-reachtruck.html

Liu, W.-Y., Lin, C.-C., Chiu, C.-R., Tsao, Y.-S., & Wang, Q. (2014). Minimizing the Carbon Footprint for the Time-Dependent Heterogeneous-Fleet Vehicle Routing Problem with Alternative Paths. *Sustainability*, *6*, 4658–4684. https://doi.org/10.3390/su6074658

Lone, S., Fàvero, I., Quaglieri, L., & Packiaraja, S. (2019). *European Ecommerce report*. https://www.ecommerce-europe.eu/wp-content/uploads/2019/07/European_Ecommerce_report_2019_freeFinal-version.pdf

Mahmoudi, M., & Zhou, X. (2016). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations. *Transportation Research Part B: Methodological*, *89*, 19–42. https://doi.org/10.1016/j.trb.2016.03.009

mastercard. (2017). *Masterindex 2017: Pan-European e-commerce and new payment trends*.

Mellahi, K., & Johnson, M. (2000). Does it pay to be a first mover in e.commerce? The case of Amazon.com. *Management Decision*, *38*(7), 445–452. https://doi.org/10.1108/00251740010373458

Nalivajevs, O., & Karapetyan, D. (2019). Conditional Markov Chain Search for the Generalised Travelling Salesman Problem for Warehouse Order Picking. *2019 11th Computer Science and Electronic Engineering (CEEC)*, 75–78. https://doi.org/10.1109/CEEC47804.2019.8974324

Öztürkoğlu, Ö., & Hoser, D. (2019). A discrete cross aisle design model for order-picking warehouses. *European Journal of Operational Research*, *275*(2), 411–430. https://doi.org/10.1016/j.ejor.2018.11.037

Petersen, C. (2009). An evaluation of order picking policies for mail order companies. *Production and Operations Management*, *9*, 319–335. https://doi.org/10.1111/j.1937-5956.2000.tb00461.x

Petersen, C. G. (2002). Considerations in order picking zone configuration. *International Journal of Operations & Production Management*, *22*(7), 793–805. https://doi.org/10.1108/01443570210433553

Pintea, C.-M., Chira, C., & Dumitrescu, D. (2011). Sensitive Ants in Solving the Generalized Vehicle Routing Problem. *International Journal of Computers, Communications and Control*, *6*(4), 731–738. http://dx.doi.org/10.15837/ijccc.2011.4.2098

Pintea, C.-M., Pop, P. C., & Chira, C. (2017). The generalized traveling salesman problem solved with ant algorithms. *Complex Adaptive Systems Modeling*, *5*(1), 8. https://doi.org/10.1186/s40294-017-0048-9

Pop, P. C., Matei, O., & Sitar, C. P. (2013). An improved hybrid algorithm for solving the generalized vehicle routing problem. *Neurocomputing*, *109*, 76–83. https://doi.org/10.1016/j.neucom.2012.03.032

Pop, P., Matei, O., & Valean, H. (2011). An Efficient Hybrid Soft Computing Approach to the Generalized Vehicle Routing Problem. In E. Corchado, V. Snášel, J. Sedano, A. E. Hassanien, J. L. Calvo, & D. Ślęzak (Red.), *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011* (pp. 281–289). Springer. https://doi.org/10.1007/978-3-642-19644-7_30

Roodbergen, K. J., & De Koster, R. (2001). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, *39*(9), 1865–1883. https://doi.org/10.1080/00207540110028128

Roodbergen, K., & Vis, I. F. A. (2006). A model for warehouse layout. *IIE Transactions*, *38*(10), 799–811. https://doi.org/10.1080/07408170500494566

Russell, S. H. (2011). Supply Chain Management: More Than Integrated Logistics. *Air Force Journal of Logistics; Gunter AFS*, *35*(3/4), 88–96. http://search.proquest.com/docview/1430897272/abstract/FC5C05EEC0CC4E50PQ/1

Rust, R. T., & Kannan, P. K. (2003). E-service: A new paradigm for business in the electronic environment. *Communications of the ACM*, *46*(6), 36–42. https://doi.org/10.1145/777313.777336

Scholz, A., Henn, S., Stuhlmann, M., & Wäscher, G. (2016). A new mathematical programming formulation for the Single-Picker Routing Problem. *European Journal of Operational Research*, *253*(1), 68–84. https://doi.org/10.1016/j.ejor.2016.02.018

Smith, S. L., & Imeson, F. (2017). GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem. *Computers & Operations Research*, *87*, 1–19. https://doi.org/10.1016/j.cor.2017.05.010

Statista. (2020). *E-commerce share of total retail sales*. Statista. https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/

Sundar, K., & Rathinam, S. (2016). Generalized multiple depot traveling salesmen problem—Polyhedral study and exact algorithm. *Computers & Operations Research*, *70*, 39–55. https://doi.org/10.1016/j.cor.2015.12.014

Tabitha, C. (2019). *2019 ecommerce in review: Consumer insights*. https://www.digitalcommerce360.com/2019/12/23/2019-ecommerce-in-review-consumer-insights/

Theys, C., Bräysy, O., Dullaert, W., & Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, *200*(3), 755–763. https://doi.org/10.1016/j.ejor.2009.01.036

Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities Planning*. John Wiley & Sons.

Van Dale Uitgevers. (2020). *Van Dale* [Woordenboek]. https://www.vandale.nl/gratis-woordenboek/engels-nederlands/vertaling/e-commerce

Van Dooren, P. (2020, oktober 14). Prijzen logistiek vastgoed stijgen fors door schaarste. *Flows*. https://www.flows.be/nl/logistics/logistiek-vastgoed-fors-duurder-ondanks-coronacrisis

van Gils, T., Braekers, K., Ramaekers, K., Depaire, B., & Caris, A. (2016). Improving Order Picking Efficiency by Analyzing Combinations of Storage, Batching, Zoning, and Routing Policies. In A. Paias, M. Ruthmair, & S. Voß (Red.), *Computational Logistics* (Vol. 9855, pp. 427–442). Springer International Publishing. https://doi.org/10.1007/978-3-319-44896-1_28

van Gils, T., Caris, A., Ramaekers, K., & Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, *277*(3), 814–830. https://doi.org/10.1016/j.ejor.2019.03.012

van Gils, T., Ramaekers, K., Braekers, K., Depaire, B., & Caris, A. (2018). Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions. *International Journal of Production Economics*, *197*, 243–261. https://doi.org/10.1016/j.ijpe.2017.11.021

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, *95*, 139–150. https://doi.org/10.1016/j.cor.2018.03.012

Weidinger, F., & Boysen, N. (2018). Scattered Storage: How to Distribute Stock Keeping Units All Around a Mixed-Shelves Warehouse. *Transportation Science*, *52*(6), 1412–1427. https://doi.org/10.1287/trsc.2017.0779

Weidinger, F., Boysen, N., & Schneider, M. (2018). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, *274*(2), 501–515. https://doi.org/10.1016/j.ejor.2018.10.021

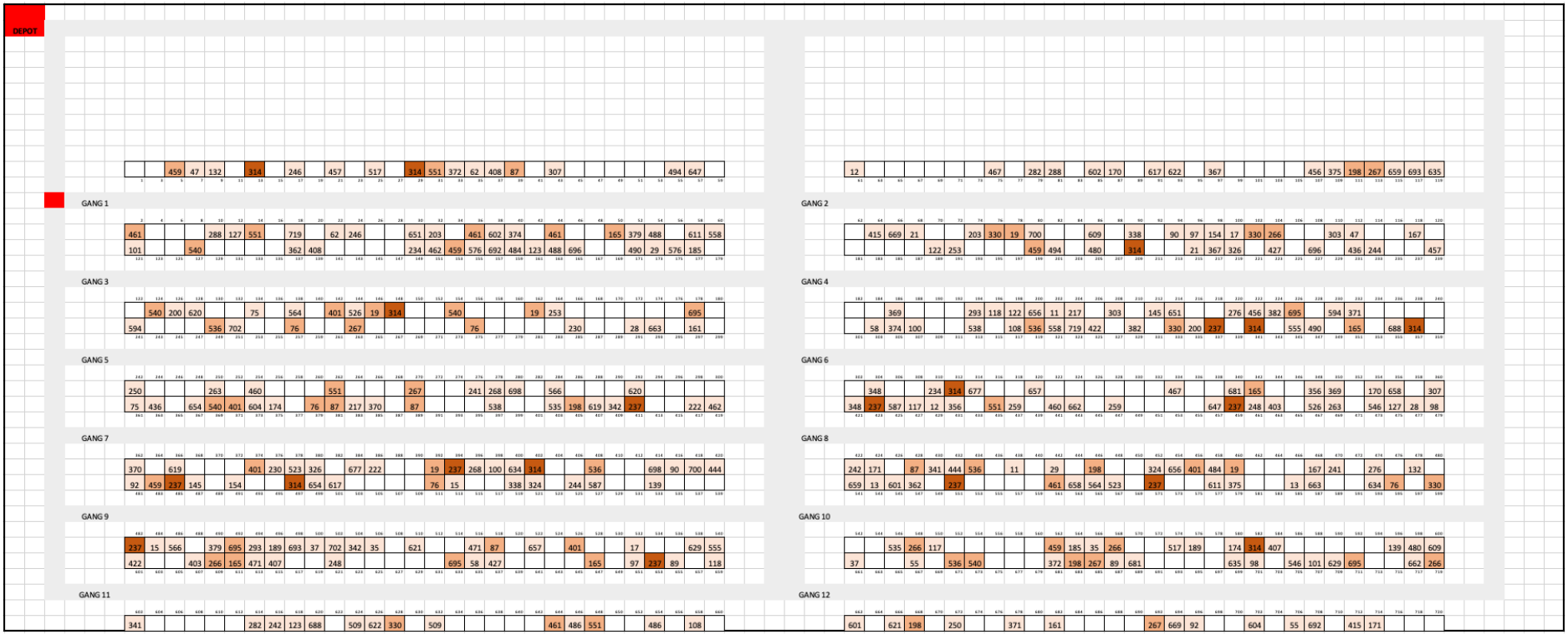
Wolf, E. G., Eckman, C. F., & Woolf, A. J. (2018). *Warehouse rack space optimization* (United States Patent Nr. US10019693B2). https://patents.google.com/patent/US10019693B2/en

Yu, Y., Koster, R. B. M. de, & Guo, X. (2015). Class-Based Storage with a Finite Number of Items: Using More Classes is not Always Better. *Production and Operations Management*, *24*(8), 1235–1247. https://doi.org/10.1111/poms.12334

6 Bijlagen

6.1 De voorbeeld magazijnen

Het ingevulde voorbeeld magazijn voor de datagroep met maximaal vier orderlijnen:

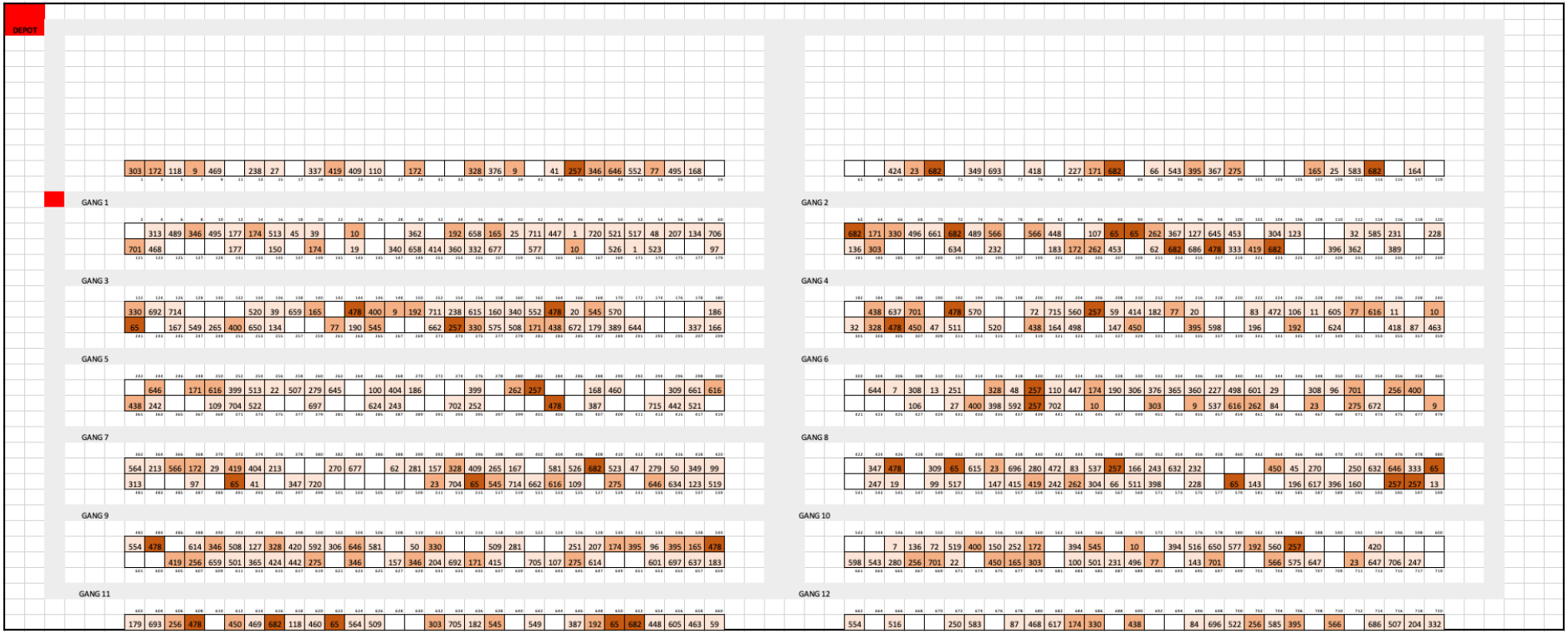


Figuur 31: Het ingevulde voorbeeld magazijn voor de datagroep met maximaal vier orderlijnen.

Het ingevulde voorbeeld magazijn voor de datagroep met maximaal acht orderlijnen:

Figuur 32: Het ingevulde voorbeeld magazijn voor de datagroep met maximaal acht orderlijnen.

Het ingevulde voorbeeld magazijn voor de datagroep met maximaal twaalf orderlijnen:



Figuur 33: Het ingevulde voorbeeld magazijn voor de datagroep met maximaal twaalf orderlijnen.

6.2 De oplossingsprestaties per picklijst-groep

6.2.1 De prioriteitsheuristieken met nieuwe data

SinglePosition				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	40	66,67%	3,46%	61,35%
8 Orderlijnen	44	73,33%	3,31%	35,21%
12 Orderlijnen	30	50%	6,25%	53,40%
Totaal	**114**	**63,33%**	**4,34%**	**61,35%**

Tabel 22: De oplossingsprestaties voor SinglePosition met de nieuwe dataset.

MinMin				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	45	75%	1,70%	20,71%
8 Orderlijnen	46	76,67%	1,67%	15,60%
12 Orderlijnen	48	80%	1,71%	35,74%
Totaal	**139**	**77,22%**	**1,69%**	**35,74%**

Tabel 23: De oplossingsprestaties voor MinMin met de nieuwe dataset.

MinMax				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	43	71,67%	1,90%	16,62%
8 Orderlijnen	37	61,67%	4,33%	29,08%
12 Orderlijnen	40	66,67%	4,16%	35,76%
Totaal	**120**	**66,67%**	**3,47%**	**35,76%**

Tabel 24: De oplossingsprestaties voor MinMax met de nieuwe dataset.

6.2.2 FIFO-batching

SinglePosition				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	15	50%	3,98%	29,83%
8 Orderlijnen	13	43,33%	5,91%	28,07%
12 Orderlijnen	11	36,67%	5,95%	28,46%
Totaal	**39**	**43,33%**	**5,28%**	**29,83%**

Tabel 25: De oplossingsprestaties voor SinglePosition met de data na FIFO-batching.

MinMin				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	19	63,33%	2,21%	11,34%
8 Orderlijnen	23	76,67%	1,48%	13,66%
12 Orderlijnen	23	76,67%	2,61%	22,78%
Totaal	**65**	**72,22%**	**2,10%**	**22,78%**

Tabel 26: De oplossingsprestaties voor MinMin met de data na FIFO-batching.

MinMax				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	13	43,33%	6,09%	30,28%
8 Orderlijnen	8	26,67%	8,79%	55,98%
12 Orderlijnen	8	26,67%	9,32%	36,25%
Totaal	**29**	**32,22%**	**8,06%**	**55,98%**

Tabel 27: De oplossingsprestaties voor MinMax met de data na FIFO-batching.

6.2.3 SinglePosition batching

SinglePosition				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	11	36,67%	7,71%	30,28%
8 Orderlijnen	12	40%	7,45%	30,78%
12 Orderlijnen	13	43,33%	7,57%	36,70%
Totaal	**36**	**40%**	**7,58%**	**36,70%**

Tabel 28: De oplossingsprestaties voor SinglePosition met de data na SinglePosition batching.

MinMin				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	14	46,67%	8,34%	43,09%
8 Orderlijnen	22	73,33%	2,36%	15,48%
12 Orderlijnen	22	73,33%	2,02%	22,78%
Totaal	**58**	**64,44%**	**4,24%**	**43,09%**

Tabel 29: De oplossingsprestaties voor MinMin met de data na SinglePosition batching.

MinMax				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	21	70%	2,88%	27,74%
8 Orderlijnen	8	26,67%	10,04%	55,98%
12 Orderlijnen	7	23,33%	10,02%	48,65%
Totaal	**36**	**40%**	**7,65%**	**55,98%**

Tabel 30: De oplossingsprestaties voor MinMax met de data na SinglePosition batching.

6.2.4 SinglePosition batching met een alternatieve eerste SKU

SinglePosition				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	18	60%	6,08%	43,09%
8 Orderlijnen	13	43,33%	7,24%	30,78%
12 Orderlijnen	14	46,67%	6,74%	30,84%
Totaal	**45**	**50%**	**6,75%**	**43,09%**

Tabel 31: De oplossingsprestaties voor SinglePosition met de data na SinglePosition batching met de alternatieve methode voor het selecteren van de eerste SKU.

MinMin				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	18	60%	3,56%	27,74%
8 Orderlijnen	21	70%	2,15%	15,48%
12 Orderlijnen	24	80%	1,92%	22,78%
Totaal	**63**	**70%**	**2,54%**	**27,74%**

Tabel 32: De oplossingsprestaties voor MinMin met de data na SinglePosition batching met de alternatieve methode voor het selecteren van de eerste SKU.

MinMax				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	11	36,67%	6,96%	30,28%
8 Orderlijnen	7	23,33%	10,04%	55,98%
12 Orderlijnen	9	30%	9,50%	22,78%
Totaal	**27**	**30%**	**8,84%**	**55,98%**

Tabel 33: De oplossingsprestaties voor MinMax met de data na SinglePosition batching met de alternatieve methode voor het selecteren van de eerste SKU.

6.2.5 De parallelle opslaglocatie-selectie met de onderschattende methode

Onderschatting				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	47	78,33%	3,66%	99,50%
8 Orderlijnen	45	75%	4,34%	68,50%
12 Orderlijnen	45	75%	5,56%	114%
Totaal	**137**	**76,11%**	**4,52%**	**114%**

Tabel 34: De oplossingsprestaties van de onderschattende methode.

6.2.6 De parallelle opslaglocatie-selectie met de overschattende methode

Overschatting				
	# Beste oplossing	**% Beste oplossing**	**Gemiddelde kloof**	**Maximale kloof**
4 Orderlijnen	46	76,67%	4,20%	49,61%
8 Orderlijnen	36	60%	9,16%	108,34%
12 Orderlijnen	18	30%	3,78%	51%
Totaal	**100**	**55,56%**	**5,74%**	**108,34%**

Tabel 35: De oplossingsprestaties van de overschattende methode.

6.3 De Lingo code van de parallelle opslaglocatie-selectie formulering

6.3.1 De Lingo code van de onderschattende methode

In 'Figuur 34' is een voorbeeld zichtbaar van de Lingo code voor de onderschattende methode. Hierbij werd de formulering ingevuld voor het eerste order uit de picklijst 'Figuur *21*'. Daarbij staan SKUs 92, 526, 576 en 13 op de picklijst met een gevraagde hoeveelheid van respectievelijk 2, 1, 4 en 1 eenheden.

```
! 0 = depot;
! SKU1 = 92, locaties: 481-696, gevraagde hoeveelheid 2;
! SKU2 = 526, locaties: 144-467, gevraagde hoeveelheid 1;
! SKU3 = 576, locaties: 155-175, gevraagde hoeveelheid 4;
! SKU4 = 13, locaties: 543-585, gevraagde hoeveelheid 1;
! Alle mogelijke opslaglocaties (gesorteerd): 144, 155, 175, 467, 481, 543, 585, 696;


! De doelfunctie minimaliseert de afgelegde afstand.;
MIN= 1043*Y0_144 + 1277*Y0_155 + 1667*Y0_175+ 3149*Y0_467 + 1046*Y0_481 + 2435*Y0_543 + 3254*Y0_585 + 3203*Y0_696 +
          234*Y144_155  + 624*Y144_175  + 2107*Y144_467 + 1081*Y144_481 + 1393*Y144_543 + 2212*Y144_585 + 2161*Y144_696 +
          390*Y155_175  + 1873*Y155_467 + 1315*Y155_481 + 1159*Y155_543 + 1978*Y155_585 + 1927*Y155_696 +
          1483*Y175_467 + 1705*Y175_481 + 769*Y175_543  + 1588*Y175_585 + 1537*Y175_696 +
          2392*Y467_481 + 1300*Y467_543 + 871*Y467_585  + 1210*Y467_696 +
          1390*Y481_543 + 2209*Y481_585 + 2158*Y481_696 +
          819*Y543_585  + 1066*Y543_696 +
          1105*Y585_696;


! Locaties selecteren;
! De gevraagde hoeveelheid picken rekening houdend met de beschikbare voorraad per opslaglocatie;
X481 + X696 = 2;
X481 <= 3*Z481;
X696 <= 4*Z696;

X144 + X467 = 1;
X144 <= 3*Z144;
X467 <= 4*Z467;

X155 + X175 = 4;
X155 <= 2*Z155;
X175 <= 2*Z175;

X543 + X585 = 1;
X543 <= 2*Z543;
X585 <= 4*Z585;


! Een route moet gevormd worden tussen de geselecteerde locaties;
! De route moet starten en eindigen in het depot;
Y0_144 + Y0_155 + Y0_175+ Y0_467 + Y0_481 + Y0_543 + Y0_585 + Y0_696 = 2;
! Indien een locatie geselecteerd is, dan moet er een in- en uitgaand pad zijn;
Y0_481 + Y144_481 + Y155_481 + Y175_481 + Y467_481 + Y481_543 + Y481_585 + Y481_696 = 2*Z481;
Y0_696 + Y144_696 + Y155_696 + Y175_696 + Y467_696 + Y481_696 + Y543_696 + Y585_696 = 2*Z696;
Y0_144 + Y144_155 + Y144_175 + Y144_467 + Y144_481 + Y144_543 + Y144_585 + Y144_696 = 2*Z144;
Y0_467 + Y144_467 + Y155_467 + Y175_467 + Y467_481 + Y467_543 + Y467_585 + Y467_696 = 2*Z467;
Y0_155 + Y144_155 + Y155_175 + Y155_467 + Y155_481 + Y155_543 + Y155_585 + Y155_696 = 2*Z155;
Y0_175 + Y144_175 + Y155_175 + Y175_467 + Y175_481 + Y175_543 + Y175_585 + Y175_696 = 2*Z175;
Y0_543 + Y144_543 + Y155_543 + Y175_543 + Y467_543 + Y481_543 + Y543_585 + Y543_696 = 2*Z543;
Y0_585 + Y144_585 + Y155_585 + Y175_585 + Y467_585 + Y481_585 + Y543_585 + Y585_696 = 2*Z585;


! Datatypen;
@BIN(Z144); @BIN(Z155); @BIN(Z175); @BIN(Z467); @BIN(Z481); @BIN(Z543); @BIN(Z585); @BIN(Z696);
```

Figuur 34: Een voorbeeld van de onderschattende formulering.

6.3.2 De Lingo code van de overschattende methode

In 'Figuur 35' is een voorbeeld zichtbaar van de Lingo code voor de overschattende methode. Hierbij werd de formulering ingevuld voor het eerste order uit de picklijst 'Figuur *21*'. Daarbij staan SKUs 92, 526, 576 en 13 op de picklijst met een gevraagde hoeveelheid van respectievelijk 2, 1, 4 en 1 eenheden.

```
! 0 = depot;
! SKU1 = 92, locaties: 481-696, gevraagde hoeveelheid 2;
! SKU2 = 526, locaties: 144-467, gevraagde hoeveelheid 1;
! SKU3 = 576, locaties: 155-175, gevraagde hoeveelheid 4;
! SKU4 = 13, locaties: 543-585, gevraagde hoeveelheid 1;
! Alle mogelijke opslaglocaties (gesorteerd): 144, 155, 175, 467, 481, 543, 585, 696;


! De doelfunctie minimaliseert de afgelegde afstand.;
MIN= 1043*Y0_144 + 1277*Y0_155 + 1667*Y0_175+ 3149*Y0_467 + 1046*Y0_481 + 2435*Y0_543 + 3254*Y0_585 + 3203*Y0_696 +
234*Y144_155 + 624*Y144_175 + 2107*Y144_467 + 1081*Y144_481 + 1393*Y144_543 + 2212*Y144_585 + 2161*Y144_696 +
390*Y155_175 + 1873*Y155_467 + 1315*Y155_481 + 1159*Y155_543 + 1978*Y155_585 + 1927*Y155_696 +
1483*Y175_467 + 1705*Y175_481 + 769*Y175_543 + 1588*Y175_585 + 1537*Y175_696 +
2392*Y467_481 + 1300*Y467_543 + 871*Y467_585 + 1210*Y467_696 +
1390*Y481_543 + 2209*Y481_585 + 2158*Y481_696 +
819*Y543_585 + 1066*Y543_696 +
1105*Y585_696;


! Locaties selecteren;
! De gevraagde hoeveelheid picken rekening houdend met de beschikbare voorraad per opslaglocatie;
X481 + X696 = 2;
X481 <= 3*Z481;
X696 <= 4*Z696;

X144 + X467 = 1;
X144 <= 3*Z144;
X467 <= 4*Z467;

X155 + X175 = 4;
X155 <= 2*Z155;
X175 <= 2*Z175;

X543 + X585 = 1;
X543 <= 2*Z543;
X585 <= 4*Z585;


! Alle paden tussen geselecteerde locaties worden geactiveerd;
Z144 + Z155 <= 1 + Y144_155; Z144 + Z175 <= 1 + Y144_175; Z144 + Z467 <= 1 + Y144_467; Z144 + Z481 <= 1 + Y144_481;
Z144 + Z543 <= 1 + Y144_543; Z144 + Z585 <= 1 + Y144_585; Z144 + Z696 <= 1 + Y144_696; Z155 + Z175 <= 1 + Y155_175;
Z155 + Z467 <= 1 + Y155_467; Z155 + Z481 <= 1 + Y155_481; Z155 + Z543 <= 1 + Y155_543; Z155 + Z585 <= 1 + Y155_585;
Z155 + Z696 <= 1 + Y155_696; Z175 + Z467 <= 1 + Y175_467; Z175 + Z481 <= 1 + Y175_481; Z175 + Z543 <= 1 + Y175_543;
Z175 + Z585 <= 1 + Y175_585; Z175 + Z696 <= 1 + Y175_696; Z467 + Z481 <= 1 + Y467_481; Z467 + Z543 <= 1 + Y467_543;
Z467 + Z585 <= 1 + Y467_585; Z467 + Z696 <= 1 + Y467_696; Z481 + Z543 <= 1 + Y481_543; Z481 + Z585 <= 1 + Y481_585;
Z481 + Z696 <= 1 + Y481_696; Z543 + Z585 <= 1 + Y543_585; Z543 + Z696 <= 1 + Y543_696; Z585 + Z696 <= 1 + Y585_696;


! Datatypen;
@BIN(Z144); @BIN(Z155); @BIN(Z175); @BIN(Z467); @BIN(Z481); @BIN(Z543); @BIN(Z585); @BIN(Z696);
```

Figuur 35: Een voorbeeld van de overschattende formulering.