Radboud Universiteit

Modellenpracticum
Voorjaar 2017

Royal Smit
Een voortzetting tot een efficiënter slitproces

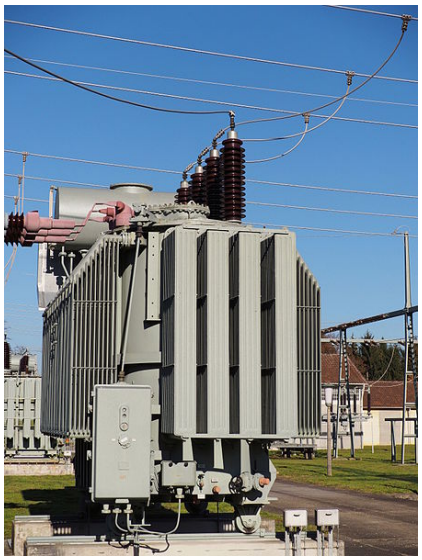
Grzegorz Baltissen
Jorrit Bastings
Rebekka Willenberg
Robin Wubs

onder begeleiding van:

Wieb Bosma en Mark Jansen

Inhoudsopgave

1 Inleiding



Figuur 1: Transformator voor de omzetting van 120kV naar 20kV
Bron: Kreuzschnabel/Wikimedia Commons, Licentie: CC-by-sa-3.0

Voor het omzetten van stroom naar een andere spanning is een transformator nodig. Denk bijvoorbeeld aan een telefoonoplader die 230 Volt omzet naar 4,5 Volt of aan een hoogspanningstransformator zoals in figuur 1. Royal Smit is een bedrijf dat hoogspanningstransformatoren op aanvraag ontwerpt en fabriceert. Het bedrijf is in deze sector de marktleider en dus de hoofdleverancier voor bijvoorbeeld energiecentrales die hun gewenste trafo niet kunnen kopen bij een ander bedrijf met een vast catalogus. Royal Smit is dus vanaf het ontwerpen tot en met het in elkaar zetten van transformatoren verantwoordelijk.

Elke transformator heeft een stalen kern. De kern van zo'n hoogspanningstransformator is te groot om uit één blok staal gemaakt te kunnen worden. Daarom stapelt Royal Smit platen blik op elkaar. De op deze manier gefabriceerde kernen lijken dan een beetje op een drietand. Zie figuur 2 voor een foto (met dank aan Marco Schieke) van een kern die bij Royal Smit geproduceerd werd. Zoals te zien is, hebben deze platen verschillende afmetingen. Deze platen worden binnen het bedrijf uit zogenaamde moederrollen geslit (gesneden).

Het slitproces

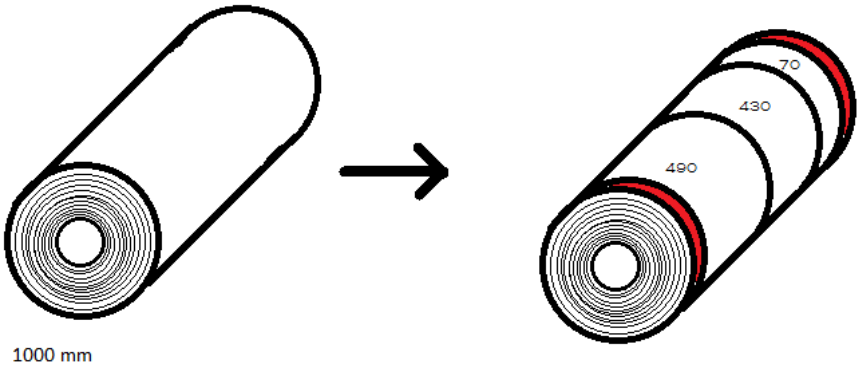
Royal Smit slit rollen blik, zogenaamde moederrollen, in subrollen om uit deze subrollen later de kern van een transformator te kunnen maken. Het slitten gaat als volgt:
Het bedrijf heeft rollen blik van verscheidene leveranciers en materialen op voorraad. Voor een order/transformatorkern moeten uit deze voorraad de gewenste moederrollen geselecteerd worden. Uit deze selectie moederrollen slit het bedrijf smallere rollen, zogenaamde orderbreedtes. En van elke orderbreedte is een bepaalde behoefte voor de transformator nodig. We hebben hiervoor een voorbeeld, zie figuur 3. De orderbreedtes/subrollen worden vervolgens in een ander proces tot losse lengtes geknipt.
Wij zijn gevraagd om kosten helpen te besparen bij Royal Smit. Dit hebben we aan de hand van drie vragen gedaan. Met als eerste: *Hoe moet je de gewenste*

Figuur 2: Laagjes blik vormen samen de kern van een transformator

breedtes uit de moederrollen slitten zodat je zo min mogelijk afval produceert? Royal Smit laat dit door een persoon uitpuzzelen en er is geen garantie dat het een goede oplossing oplevert. Graag wil Royal Smit een programma hebben dat met hoge garantie een goede slitoplossing geeft.
Verder zit veel tijd in het slitten zelf. Op dit moment worden de orderbreedtes per moederrol van groot naar klein gesorteerd en vervolgens die moederrollen achter elkaar geslit waar het patroon overeen komt. Per nieuwe breedte moeten messen nieuw ingesteld worden. Door de orderbreedtes per moederrol beter te sorteren kan er zeker meer tijd bespaart worden. Dan hebben we als tweede vraag: *Hoe moeten de orderbreedtes per moederrol en de moederrollen gesorteerd worden zodat er zo min mogelijk messen gewisseld moeten worden?*
Ook is er bij het bedrijf een zekere willekeur in wat voor soort moederrollen het ontvangt bij het inkopen. Hierbij hebben we als laatste vraag: *Kan het bedrijf kosten besparen door de moederrollen specifieker in te kopen, en zo ja, hoe?*
Deze vragen zullen we in het verslag beschouwen.

Het project bij Royal Smit

Wij zijn de tweede groep studenten die voor het vak 'Modellenpracticum' dit vraagstuk onderzoekt. Vorig jaar al werd het project in drie fases opgedeeld:



1000 mm
4210 kg

batchnummer & gewicht moederrol	messenstand (mm)	ordernummer	orderbreedte	benodigd gewicht (kg)	gesneden gewicht (kg)
C1000P0889 (4210 kg)	5/490/430/70/5	312000730	HE150490a	3724.3	2189.2
		312000730	HE150430a	1489.44	1852.4
			Courante offcut		294.7
			Randen		23.9

Figuur 3: slitting

1. Prototype programma voor slitoplossingen.
2. Eisen aan het definitieve programma.
3. Optimalisatie bij het inkopen

Aanpak en resultaten van vorig jaar

Vorig jaar schreef het groepje eerst een brute-force programma om daadwerkelijk de beste combinatie van orderbreedtes voor de moederrollen te vinden. Dit werkte echter niet omdat er te veel data is en het programma er te lang over zou doen. Uiteindelijk hebben ze een algoritme gekozen dat wel een oplossing geeft. Hiermee hadden ze dus een werkend prototype en fase 1 afgerond. Hoe het bij hen precies verliep is te lezen in hun verslag.

Aanpak en resultaten van dit jaar

Door Royal Smit werd gevraagd om een aantal nieuwe randvoorwaarden te implementeren. Dit pakten we aan door het oude programma aan te passen. Hierbij merkten we op dat er in het oude programma nog een aantal bugs zaten. Zo werd bijvoorbeeld niet van elke order de benodigde hoeveelheid geslit als deze niet in een moederrol past. We haalden de bugs uit het programma en hebben de nieuwe randvoorwaarden geimplementeerd zodat we uiteindelijk een

programma kregen dat een juiste slitoplossing geeft. Op een gegeven voorraad moederrollen en orders genereerde het programma een slitoplossing en deze konden we vergelijken met de slitoplossing van Marco Schieke. Het bleek dat ons programma een moederrol meer nodig had. Vervolgens vroegen we ons af welke alternatieve algoritmen een beter resultaat zouden kunnen geven. Aangezien de code inmiddels erg ingewikkeld en ongestructureerd was hebben we deze niet meer kunnen implementeren. De concepten zullen in dit verslag uitgelegd worden zodat ze door een volgend groepje geimplementeerd zouden kunnen worden. We zullen ingaan op object-georienteerd programmeren, lineair programmeren en boomdoorzoekende algoritmen.
Onafhankelijk van het indelen van de moederrollen zijn er kosten te besparen door een efficiënt gebruik van de slitmachine. Door de orderbreedtes per moederrol in een slimme volgorde te zetten en vervolgens de moederrollen in een slimme volgorde te slitten moeten minder messen gewisseld worden. Ook dit probleem bleek in het algemene geval NP-hard te zijn. Dit wordt duidelijk in het hoofdstuk over het Handelsreizigersprobleem. Zelfs voor een gemiddelde slitoplossing duurt het te lang om alle gevallen af te gaan. Omdat we veel ideeën hebben maar niet meer de tijd om ze om te zetten hebben we ons als doel gezet om het zo nauwkeurig mogelijk op te schrijven. Dan kan volgend jaar een nieuw groepje onze concepten verder uitwerken.

2 Het huidige programma

Bij de eerste afspraak met Royal Smit werd gevraagd een programma te maken dat laat zien dat het probleem überhaupt door een computer opgelost kan worden. Dat is vorig jaar gebeurd. Dit jaar zijn een aantal nieuwe randvoorwaarden erbij gekomen zodat het oude programma out-of-date was. Door in het programma van vorig jaar een aantal dingen aan te passen hebben we een nieuwe versie kunnen maken die ook aan de nieuwe randvoorwaarden van Royal Smit voldoet. Het kostte ons veel tijd om te snappen wat er in het programma van vorig jaar gebeurde. Daarom kiezen we ervoor de grote lijnen van het programma uit te leggen.

2.1 Grote lijnen

Het programma bestaat uit een aantal documenten, die ingelezen worden, de code en een output document; de slitoplossing. Als de code uitgevoerd wordt, worden eerst het bestand met de voorraad moederrollen en de benodigde orderbreedtes van een order ingelezen. Uiteindelijk willen we als output weten welke orderbreedtes we het beste uit welke moederrollen kunnen slitten. Zowel de moederrollen als ook de orderbreedtes verschillen in gewicht en lengte. Om orderbreedtes, en moederrollen, met elkaar te kunnen vergelijken werd vorig jaar al ervoor gekozen het $\frac{gewicht}{breedte}$ te beschouwen. Het idee van het algoritme is hetzelfde als vorig jaar: De moederrollen worden oplopend en de orderbreedtes aflopend op $\frac{gewicht}{breedte}$ gesorteerd. Vervolgens wordt de kleinste (t.o.v. $\frac{gewicht}{breedte}$)

moederrol gezocht waar de grootste (t.o.v. $\frac{gewicht}{breedte}$) orderbreedte in past. Gebaseerd op dit algoritme wordt een slitoplossing gemaakt. Deze wordt in de output gegeven. Een voorbeeld van een slitoplossing is te zien in figuur 4.

Hieronder volgt een beschrijving van de randvoorwaarden die voor het programma gelden. De moederrollen worden besteld bij verschillende leveranciers. Als een order ingedeeld wordt moet dat uit moederrollen van één leverancier zijn. Daarom wordt per leverancier gekeken wat een mogelijke indeling van de moederrollen is en de beste slitoplossing wordt als output gegeven. Meestal is het niet mogelijk om de moederrollen qua breedte helemaal te gebruiken. Omdat een zusteronderneming van Royal Smit smallere moederrollen nodig heeft is het mogelijk om de resten van de moederrollen van de leverancier Postco door te verkopen. Hiervoor moeten deze in rollen met breedte 130, 140, 150, 160 of 170mm geslit worden. Deze breedtes worden in het programma standaardbreedtes genoemd.
Verder moeten aan de rechter en linkerkant van de moederrollen randen van minimaal 5mm geslit worden. Dit komt omdat de moederrollen vaak aan de randen beschadigd zijn en die delen uiteraard niet voor de transformatoren gebruikt mogen worden. In het programma is het zodanig geimplementeerd dat de rand standaard op 5 mm staat. Deze kan aangepast worden door dit in een extra kolom aan te geven (zie bestand moederrollen).
Als de moederrollen in smalle stukken geslit zijn worden deze subrollen weer opgewikkelt tot rollen. Dit is alleen maar mogelijk als de subrollen tenminste 50mm breed zijn. Daarom is een eis aan het programma dat de offcuts, dus datgene van de rol wat niet voor een order benodigd wordt, tenminste 50mm breed is.
Verder worden offcuts die smaller dan 30 mm zijn als rand geslit. Dat wil zeggen dat het zodanig opgesplitst wordt dat de randen (inclusief de standaard 5 mm) maximaal 20 mm breedt zijn. De orderbreedtes worden in afstanden van 10 mm geslit. Een offcut van 40 mm wordt in het programma als het minst efficient beschouwd en komt dus alleen in een slitoplossing terug als er geen andere slitoplossing mogelijk is.

3 Architectuur voor een tweede versie van het programma

Het programma dat vorig jaar was geprogrammeerd kan met hakken over de sloot een prototype genoemd worden. Als een conceptversie heeft het geïllustreerd dat het mogelijk is om een oplossing door een computer te laten berekenen. Maar in het door ons uitgewerkte programma ontbreken nog steeds belangrijke programmeerprincipes waardoor het niet mogelijk is het verder uit te bouwen. Dat inzicht onstond geleidelijk. Pas halverwege het project snapten we waar het probleem lag. Het inlezen van de voorraad was t.o.v. vorig haar verouderd. Dat was een van de eerste dingen die we moesten aanpassen. Er waren toen al signalen, dat het programma van vorig jaar niet per se de beste code was. Met-

batchnummer & gewicht moederrol	messenstand (mm)	ordernummer	orderbreedte	benodigd gewicht (kg)	gesneden gewicht (kg)	orders en standaardbreedtes (in %)	offcut (in %)	randen (in %)
C1000P0838 (1699 kg)	10/440/360/170/20	312000730	HE150440a	2237	747.56			
		312000730	HE150360a	1654	611.64			
			Standaardbreedte (170mm)		288.83			
			Randen		50.97			
						97%	0%	3%
C1000P0891 (1860 kg)	5/980/15	312000730	HE150980a	64283	1822.8			
			Randen		37.2			
						98%	0%	2%
C1000P0879 (2290 kg)	20/960/20	312000730	HE150960a	5311	2198.4			
			Randen		91.6			
						96%	0%	4%
C1000P0907 (2390 kg)	5/920/70/5	312000730	HE150920a	8336	2198.8			
			Courante offcut (70mm)		167.3			
			Randen		23.9			
						92%	7%	1%
C1000P0895 (2460 kg)	5/720/140/130/5	312000730	HE150720a	3644	1771.2			
			Standaardbreedte (140mm)		344.4			
			Standaardbreedte (130mm)		319.8			
			Randen		24.6			
						99%	0%	1%
C1000P0899 (4880 kg)	5/980/15	312000730	HE150980a	4777.4	4782.4			
			Randen		97.6			
						97.90%	0%	2%
totale efficientie zonder standaardbreedtes	0.868302							
totale efficientie met standaardbreedtes	0.933895							
Planning:								
ordernummer	duur	leverdatum						
312000730	63 uur en 20 minuten	02.06.2017						

Figuur 4: Voorbeeld van een Slitoplossing

tertijd begon de code te 'kraken' onder zijn eigen gewicht bij de toevoegingen die we deden en uiteindelijk lukte het ons niet meer de laatste toevoegingen te implementeren. Daarom raden we het volgende groepje sterk aan nieuwe code te schrijven en niet tijd kwijt te raken door het huidige programma proberen te begrijpen.
Er is op te merken dat we absoluut geen afbreuk willen doen aan wat het vorig groepje aan werk en inzet heeft verricht. Dit project heeft namelijk een incrementeel karakter en wij willen onze bevindingen presenteren en jullie overtuigen hoe de programma dan wel geïmplementeerd dient te worden te worden.

Aanbevelingen in licht van vakken die gevolgd zijn door ons

Dit is een programmeerintesief project en dit was al bij ons bekend bij het kiezen van het project. In het vorig groepje was er maar één persoon die kon programmeren, met kennis van Imperatief Programmeren 1 en 2 (IP). En nog tijdens het project deed zij Object-Oriëntatie (OO). Zij kon daarom niet de verworven kennis van OO in het project implementeren. Twee van ons groepje hadden reeds OO gevolgd en van die twee had ook een het vak Datamining gevolgd.
We denken dat de kennis van OO zeer nuttig is voor dit project. Anders is http://www.composingprograms.com [2] een volledig en kort referentiewerk van de programmeerprincipes voor een complex programma.

3.1 De programmeertaal

De conceptcode is in C++ geschreven. Deze taal is een voortzetting van C in de zin dat het meer programmeer-technieken aankan en dat de syntax lijkt op die van zijn voorganger. C++ is zeer goed om algoritmes op de machine te optimaliseren. En dat is wel fijn om te kunnen als de abstracte algoritmiek al is uitgewerkt. Maar omdat wij nog geen goede algoritmes hebben, is een taal waarin algoritmes makkelijk te implementeren zijn effectiever. C++ kan bijvoorbeeld object-georiënteerd programmeren, maar het is niet zo *straight-forward* zoals bij Java als je OO hebt gehad of zoals bij Python (strak meer over OO). Ons voorstel is om over te stappen naar Java of Python.

Python

Java en Python zijn gemakkelijker om ideeën in uit te drukken dan in C++. Zo komt het dat Java de standaard is in 'de industrie'. Toch zijn er twee redenen die Python geschikter maakt voor dit project ten opzichte van Java. Ten eerste, Python wordt ontwikkelt met een prioriteit op lees- en schrijfbaarheid. En totdat de code 'af' is, zal er nog vaak iets aan veranderd moeten worden. En dus zal er ook nog veel code gelezen moeten worden. Ten tweede, er is er een zeer gedegen referentiewerk voor het programmeren van complexe programma's, en die is geschreven voor Python: http://www.composingprograms.com/

3.2 Het verdelen van taken in de code

Omdat er geen OO was toegepast in het programma konden we bijvoorbeeld niet de messwissels implementeren en uitproberen. Dit kwam omdat in het programma de berekening van de slitoplossing en het uitlezen ervan tegelijk werd verricht. Als de oplossing nog niet klaar was, had het geen zin om de meswissels te sorteren. Was de oplossing klaar, dan was het uitlezen ook klaar. We zien dus dat bepaalde taken onnodig vermengd met elkaar zijn.

Conceptuele uitwerking belangrijke klassen

Globaal gezien bestaat het programma maar uit drie taken, zie ook figuur 5. De inlezer maakt twee lijsten: een van moederrollen en een van orderbehoeftes. De slitter maakt van deze twee lijsten een geordende lijst van gesneden rollen die de uitlezer kan aannemen.

Figuur 5:

4 Reflecties op de algoritmiek

In dit hoofdstuk beschouwen we drie ideeën voor het probleem: boomdoorzoekende algoritmes, het knapzak-probleem en lineair programmeren. Het huidige algoritme wordt doorgelicht op efficiëntie om daarna boomdoorzoekende algoritmes in het algemeen te beschouwen. Dan behandelen we de theorie van het knapzak probleem om daarna dit hoofdstuk af te sluiten met lineair programmeren

4.1 Het huidige algoritme

Het huidige algoritme sorteert zowel de moederrollen als de orders op $\frac{gewicht}{breedte}$. De orders aflopend en de moederrollen oplopend. Omdat de rollen een vaste dikte en dichtheid hebben, is gewicht per breedte, gpb, een maat die evenredig is met de lengte. Het idee hierachter was om zo veel mogelijk te optimaliseren op verlies in de lengte, d.w.z dat er niet veel te lange moederrollen gebruikt worden voor kleine orders. Een preciezere beschrijving halen we uit het verslag van vorig jaar:

We bekijken orderbreedte i met gewicht g_i en breedte b_i . Het idee is als volgt:

Als g_i/b_i groter is dan g/b van de laatste moederrol waar hij qua breedte in zou passen, dan kan deze orderbreedte niet in zijn geheel uit één moederrol gesneden worden. Snij orderbreedte i dan uit de eerste moederrol waarbij meer dan 40% van het gewicht gesneden wordt. Is er geen enkele moederrol waar minstens 40% uit gesneden kan worden, snijd dan orderbreedte i uit de laatste moederrol waar hij qua breedte inpast. Voeg het deel wat overblijft van de orderbreedte weer toe aan de lijst van orderbreedtes en ga door naar de volgende orderbreedte uit de lijst.

Als gi/bi niet groter is dan g/b van de laatste moederrol, snijd de orderbreedte dan uit de eerste moederrol waar die in zijn geheel in past.

Het algoritme genereert zo een oplossing als het een oplossing kan vinden. Als de oplossing een moederrol bevat die voor minder dan 50% (gemeten in de breedte) benut wordt, dan wordt deze moederrol uit de voorraadlijst weggehaald en wordt een nieuwe oplossing gezocht zonder deze moederrol. Dit herhaalt zich tot er een oplossing is die geen moederrollen bevat die voor minder dan 50% gebruikt zijn, of tot er geen oplossing meer gevonden kan worden. De oplossing die gekozen wordt is dan de eerste oplossing die geen enkele rol voor minder dan 50% gebruikt, of anders de beste tot dan toe. Hier betekent beste de oplossing met het minste materiaalverlies.

Commentaar op het huidige algoritme

Er is in de implementatie van het algoritme echter een klein probleem. In het geval dat een orderbreedte niet in zijn geheel (qua lengte) in een moederrol past, wordt volgens het algoritme eerst een deel in een moederrol gestopt. De rest wordt echter niet op de juiste manier toegevoegd aan de lijst met orders, maar wordt direct in een nieuwe rol gestopt. Het probleem wordt duidelijker met een voorbeeld:

Moederrol	Orderbreedte	benodigd kg	gesneden kg
C0960N0004	HE150360a	1778	1107.38
(2953 kg)	HE150360a	670.625	1107.38
	Courante offcut (230mm)		707.49
	Randen		30.7604
C0960N0005	HE150560a	2757	1726.08
(2959 kg)	Courante offcut (390mm)		1202.09
	Randen		30.8229
C0940N0004	HE150560a	1030.92	1729.45
(2903 kg)	Courante offcut (370mm)		1142.67
	Randen		30.883

Deze tabel is een selectie uit een slitoplossing gegenereerd door het programma. De onderste twee moederrollen worden gebruikt voor een orderbreedte van 560mm, en hebben een offcut van respectievelijk 390 en 370 mm. De bovenste moederrol wordt gebruikt voor een order van 360mm. Deze order was te groot (lang) voor de moederrol, en is dus gedeeltelijk er in gestopt. Hierna is direct het restant erbij gestopt, in plaats van de order van 560mm die op dat moment een groter gewicht per breedte had dan het restant van de order van 360mm. Het is duidelijk te zien dat dit niet efficiënt gebruik maakt van de rollen, want de complete order HE150360a kan naast de order HE150560a gestopt worden in de moederrollen C0940N0004 en C0960N0005, wat een complete moederrol bespaart.

Naast dit kleine probleem in de implementatie zijn er ook tegenargumenten voor het algoritme zelf. Een groot nadeel van dit algoritme is dat het erg sterk afhankelijk is van de eerste paar keuzes. De enige situatie waarin het algoritme meerdere oplossingen bekijkt is als een oplossing een moederrol voor minder dan 50% gebruikt. Deze moederrol is op een gegeven moment gekozen door het algoritme, en wordt dan in een nieuwe iteratie van het algoritme weggegooid waardoor er vanaf dat punt andere keuzes gemaakt worden. Maar alle keuzes die het algoritme heeft gemaakt voordat die v̈erkeerdem̈oederrol gekozen werd, zullen ook bij de volgende iteraties gemaakt worden. Dit wordt duidelijker in paragraaf 4.2 als het algoritme vanuit boomdoorzoekend perspectief beschouwd wordt.

Verder kunnen er vraagtekens gezet worden bij de aanname dat er vooral valt te besparen op verlies in de lengte. Zonder de mogelijkheid rollen halverwege af te knippen mee te nemen, zoals verderop toegelicht, is het inderdaad zo dat er veel materiaalverlies is in de lengte is. De vraag is alleen of dit significant te verminderen valt. Bij het testen van het programma op een lijst met orders bleek het programma vaak een erg groot deel (i.e. $> 80\%$ van een voorraad) van de rollen nodig te hebben, waardoor de grote rollen toch gebruikt worden, alleen worden ze later gekozen. Zoals het voorbeeld hierboven al aangeeft valt er echter wel nog te besparen door orders op een slimme manier naast elkaar te zetten waardoor er minder rollen nodig zijn.

Het is niet goed mogelijk om het optimaliseren op lengte direct te vergelijken met het optimaliseren op zo min mogelijk rollen te gebruiken omdat het programma alleen op lengte kan optimaliseren. We vermoeden echter dat het in de huidige situatie efficiënter is om te optimaliseren om zo min mogelijk rollen te gebruiken, omdat er bij Royal Smit efficiënt wordt ingekocht en er dus relatief weinig voorraad is. Het optimaliseren op lengte wordt pas efficiënt als de voorraad relatief groot is, waardoor er voor elke order een zo klein mogelijke rol gekozen kan worden. Met de huidige voorraden worden te grote (lange) rollen gebruikt omdat er geen kleinere geschikte rollen meer zijn.

Het optimaliseren op basis van lengte wordt nog minder interessant als we de mogelijkheid om moederrollen halverwege af te knippen introduceren. Het

huidige algoritme gaat er van uit dat een moederrol altijd volledig wordt geslit. Als er bijvoorbeeld een order is van 900mm breed en 1000kg en een moederrol waar 2500kg aan 900mm breedte op past, dan wordt er ook 2500kg geslit. Als het slitproces eerder wordt gestopt wordt er 1000kg geslit zoals nodig, en is er nog een ruim stuk moederrol over voor andere orders. Dit zorgt voor minder afval in de lengte, waardoor er minder op te optimaliseren valt door juiste moederrolkeuze.

De punten hierboven doen vermoeden dat er geschiktere algoritmes te vinden zijn voor dit probleem. In de volgende paragrafen onderzoeken we een paar van die mogelijkheden.

4.2 Een model voor boomdoorzoekende algoritmes

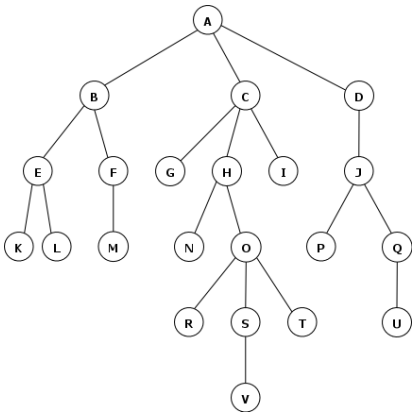
Zie eerst figuur 6, dit is wat we een boom noemen. Voor een casus als de onze heb je een zekere beginpositie, hier knoop A, waarin de moederrollen ongesneden zijn. Voor een orderbreedte zijn meerdere rollen te kiezen. En als eenmaal een rol gekozen is, heb je een zekere toestand en een bijbehorende knoop. De verschillende keuzes in moederrollen geven meerdere scenario's die voor te stellen zijn als verschillende takken naar beneden. Het heet ook wel dat een knoop kinderen heeft. Een blad is knoop zonder kinderen en is daaarom een toestand waar niet nog verder iets gekozen kan worden.

voorbeeld: Een blad in het slitproces is een toestand waarin geen moederrollen meer zijn om orders in te slitten zijn of waarin de orders juist wel zijn opgeslit.

Een boom is daarmee een uitstekende representatie van de toestandsruimte van het probleem. En van de bladeren/eindtoestanden wil je nagaan of je gewenste oplossing er tussen zit. Stel dat figuur 6 ons toestandsruimte beschrijft met de meest ideale toestand in blad M, maar dat wij dat nog niet van de boom weten dat daar de ideale toestand zit. Met een boomdoerzoekend algoritme zou je M willen tegenkomen en ervan nagaan dat hij de gewenste oplossing is.

Ongesnoeide brute-force is het simpelste boomdoorzoekende algoritme dat **alle** bladeren afgaat. Door de rijkheid aan keuzes in ons slitprobleem heeft elke tak ontiegelijk veel kinderen, en de boom is diep omdat er ook veel keuzes gemaakt moeten worden. Dit komt allemaal omdat er veel moederrollen zijn en veel orders die ingeslit moeten worden. Dit leidt ertoe dat om de hele boom af te gaan langer langer gaat duren dan de leeftijd van het universum. Dit is ook precies waar het vorig groepje achterkwam voordat het de huidige algoritme ontwikkelde.

Het huidige algoritme is ook een boomdoorzoekende algoritme. We hebben al het algoritme een paragraaf eerder vanuit een heuristisch oogpunt bekeken. Hier beschouwen we hem vanuit het boomdoorzoekende perspectief. Zie figuur 7. Het huidige algoritme herziet niet de vroeg gemaakte keuzes door het stopcriterium dat het nu hanteert. Als knoop 4a staat voor de toestand direct na het kiezen van de moederrol die uiteindelijk voor minder dan 50% gebruikt werd, dan wordt



Figuur 6: Figuur: Tsja, het lijkt ergens op een boom

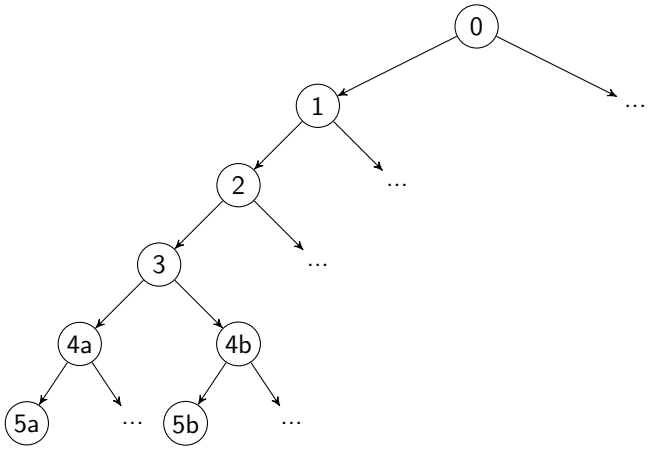
in volgende instanties alleen die keuze en de daarop volgende keuzes veranderd. Zo heb je maar een heel klein deel van alle bladeren beschouwd. Een zwakker stopcriterium zal het afgaan van de boom langer laten doorgaan. Maar er dient opgemerkt te worden dat het niet een betere oplossing hoeft te leveren als je een slechte heuristiek hebt en waardeloze bladeren afgaat.

Depth-first search en breadth-first search

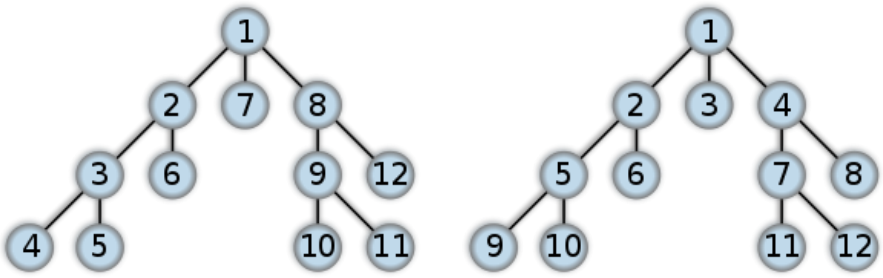
Het doorzoeken van de boom kan op vele manieren gebeuren. Depth-first search (DFS) en breadth-first search (BFS) hebben een verschillende prioriteit in de volgorde. En figuur 8 geeft dit verschil goed weer. In deze stackoverflowvraag wordt dieper op de voor- en nadelen ingegaan:
https://stackoverflow.com/questions/687731/breadth-first-vs-depth-first [1]. Wij raden aan om tussenoplossingen te berekenen met BFS om vervolgens de diepte in te duiken omdat de boom eenmaal diep is als er veel orders zijn. Zie ook [3] en [4] voor de pseudo-code.

Wat nu?

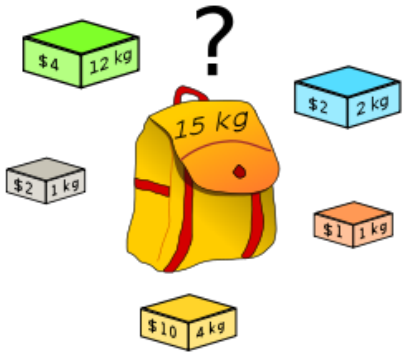
Je wil een BFS-achtig algoritme implementeren in combinatie met een DFS-achtig algoritme. Voor beide onderdelen wil je de vertakkingen afgaan die het ook waard zijn. Om dat goed te kunnen zal domeinspecifieke kennis je helpen. Dit begint met een idee te hebben van wat voor soort moederrollen er zijn en wat gebruikelijke orderbehoeftes zijn.
Voorbeeld (1): Er zijn orderbreedtes waar, ongeacht de moederrol, geen andere orderbreedte naast past. Beschouw zo'n orderbreedte. De smalste moederrol waarin deze hoeft niet per se de het beste te zijn. Het zou kunnen dat een bepaalde combinatie van orderbreedtes perfect in deze moederrol past en in bredere moederrollen een niet-toelaatbara off-cut zou vereisen. Pas BFS toe en sla tussenoplossingen op waarin je deze brede orderbreedte in meerdere moe-



Figuur 7:



Figuur 8: Bron: Alexander Drichel, Licentie: CC-BY-3.0



Figuur 9: Knapzakprobleem

derrollen plaatst.
Voorbeeld (2): De volgorde van slitten is te commuteren. Neem bijvoorbeeld een orderbreedte (900mm : 3500 kg) en twee moederrollen a:=(1000mm: 3000 kg) en b:=(1000mm: 2000 kg). Indien je (a,b) beschouwt, hoef je (b,a) niet meer te beschouwen. Ga na dat halverwege knippen deze commutativiteit niet teniet doet ("halverwege" in de zin dat een moederrol niet volledig doorgeslit wordt om de resterende hoeveelheid op een andere manier te slitten).

4.3 Het knapzak probleem

Het knapzak probleem is een heel oud probleem waar men al sinds het begin van de mensheid tegen aan liep. Het traditionele verhaal rond het knapzak probleem is als volgt. Je wilt op reis gaan maar hebt enkel een knapzak tot je beschikking. Je zou graag meer spullen mee willen nemen dan dat er in de knapzak passen. De spullen die je mee wilt nemen hebben allemaal een waarde en een gewicht. De vraag die je je zelf nu stelt is: Welke combinatie van spullen kan ik het beste meenemen in mijn knapzak? Dit probleem hangt af van een aantal factoren; het aantal knapzakken, de capaciteiten van de knapzakken, de waarde van de objecten en het gewicht van de objecten. In dit hoofdstuk zullen we een aantal situaties van het knapzak probleem behandelen. Het doel van dit hoofdstuk is om aan de hand van het knapzak probleem een beter beeld te krijgen van het optimalisatie probleem van Royal Smit. We zullen eerst het verband uitleggen tussen het knapzak probleem en het slitproces van Royal Smit en daarna een aantal voorbeelden geven van het knapzak probleem. Zo zien we bijvoorbeeld in het knapzakprobleem in Figuur 9 dat als elk object oneindig vaak voorkomt we drie gele en drie grijze mee willen nemen. Als elk object maar een keer voorkomt dan willen we alle objecten behalve de rode meenemen. We zien hier dus bijvoorbeeld dat er al twee verschillende problemen zijn en het belangrijk is dat we het probleem goed definiëren.

Het verband tussen het knapzak probleem en het slitproces van Royal Smit

In deze paragraaf zullen we het verband uitleggen tussen het knapzak probleem en het slitproces van Royal Smit.

De knapzakken
In het slitproces van Royal Smit kunnen we de moederrollen beschouwen als de knapzakken. Elke moederrol heeft een breedte en een lengte, deze noemen we de maximale capaciteiten van de moederrollen. Dit zou betekenen dat we voor elke knapzak een maximale inhoud en gewicht hebben. De maximale inhoud is de breedte van de moederrol en het maximaal gewicht is de lengte (gpb) van de moederrol.

De objecten
We nemen in het slitproces van Royal Smit als objecten de orderbreedtes. Elke orderbreedte heeft ook twee variabelen, de breedte en een lengte.

We kunnen nu het slitproces beschrijven als een bijbehorend knapzak probleem. Het knapzak probleem is alleen iets anders dan in de inleiding. In de inleiding vroegen we ons af wat de maximale waarde van objecten is die we kunnen meenemen in de knapzakken. Bij het slitproces van Royal Smit vragen we ons af wat is de 'beste' keuze van knapzakken zodanig dat we alle objecten mee kunnen nemen. Waarbij het niet meteen duidelijk is wat we hier met 'beste' bedoelen. Hier zullen we later op terug komen. Om het slitproces van Royal Smit goed te definiëren zullen we eerst een aantal standaard knapzak problemen bekijken.

$[0, 1]$ knapzak probleem

Het $[0, 1]$ knapzak probleem is een knapzak probleem met één knapzak waarbij ieder object wel of niet in de knapzak kan worden meegenomen. Wiskundig gezien komt dit neer op het volgende probleem: Gegeven een verzameling van n objecten waarbij elk object een waarde P_i heeft en een gewicht W_i. Dan zoeken we een deelverzameling objecten die de grootste waarde heeft gegeven dat de knapzak maximaal gewicht c heeft. We hebben dus:

P_i = waarde object i
W_i = gewicht object i
c = capaciteit knapzak

We zijn in dit geval op zoek naar de deelverzameling die de waarde van de knapzak maximaliseert:

$$max\{\sum_{j=1}^{n} P_j X_j | \sum_{j=1}^{n} W_j X_j \leq c\}$$

Waarbij:

$$X_j = \begin{cases} 1 & \text{als j in de knapzak;} \\ 0 & \text{als j niet in de knapzak.} \end{cases}$$

Begrensde en onbegrensd knapzak probleem

Voor het begrensd knapzak probleem nemen we aan dat van elk object er een begrensde aantal is. Bij een onbegrensd knapzak probleem is voor elk object een oneindige aantal exemplaren we zoeken dus respectievelijk:

$$\max\{\sum_{j=1}^{n} P_j X_j | \sum_{j=1}^{n} W_j X_j \leq c, X_j \in (0, 1, ..., u_j)\}$$

$$\max\{\sum_{j=1}^{n} P_j X_j | \sum_{j=1}^{n} W_j X_j \leq c, X_j \in \mathbb{N}\}$$

Het multiple knapzak probleem

Een belangrijke generalisatie van het $[0, 1]$ knapzak probleem is het multiple knapzak probleem. Hierbij hebben, we in tegenstelling tot het $[0, 1]$ knapzak probleem, meer dan één knapzak. We beschikken over m knapzakken met respectievelijk capaciteiten c_i, voor alle $i = (1, ..., m)$. Elk object kan hierbij maar in maximaal één knapzak zitten. Dit probleem is wiskundig als volgt te beschrijven, we zoeken:

$$max\{\sum_{i=1}^{m}\sum_{j=1}^{n} P_j X_{ij} | \sum_{j=1}^{n} W_j X_{ij} \leq c_i, \forall i = (1., , , .m) \text{ en } \forall j \ \sum_{i} X_{ij} \leq 1\}$$

Waarbij:

$$X_{ij} = \begin{cases} 1 & \text{als object j in knapzak i zit;} \\ 0 & \text{anders.} \end{cases}$$

Multiple-choice knapzak probleem

Het multiple-choice knapzak probleem is een generalisatie die de verzameling objecten in disjuncte partities verdeeld en waarbij we maximaal één object per partitie mogen nemen. We nemen hiervoor één knapzak met eindige capaciteit en delen de verzameling objecten op in disjuncte groepen $g_1, ..., g_p$. We willen weer de maximale waarde die we in de knapzak kunnen stoppen gegeven dat we uit elke groep g_j maximaal één object in de knapzak willen. Wiskundig gezien zijn we dus opzoek naar de maximale waarde van de knapzak zodanig dat:

$$max\{\sum_{j=1}^{n} P_j X_j | \sum_{j=1}^{n} W_j X_j \leq c \text{ en } \forall k = (1, ...p) \sum_{j \in g_k} X_j \leq 1\}$$

Waarbij:

$$X_j = \begin{cases} 1 & \text{als j in de knapzak;} \\ 0 & \text{als j niet in de knapzak.} \end{cases}$$

Multiple-choice multiple knapzak probleem

We kunnen het multiple choice knapzak probleem weer uitbreiden door het te combineren met het multiple knapzak probleem. We beschouwen dus een verzameling van n objecten verdeeld over p disjuncte groepen. Verder kijken we naar m knapzakken met respectievelijk capaciteiten c_i voor $i = (1, ..., m)$. We hebben ook in dit geval dat elke knapzak maximaal één object per groep mag bevatten en dat elk object maar in maximaal één knapzak mag zitten. Wiskundig willen we de maximale waarde van de knapzakken zodanig dat:

$$max\{\sum_{j=1}^{n} P_j X_{ij} | \sum_{j=1}^{n} W_j X_{ij} \leq c_i \, \forall i = (1, ..., m); \sum_{j \in g_k} X_{ij} \leq 1 \, \forall k = (1, ...p) \text{ en } \forall j \ \sum_{i} X_{ij} \leq 1\}$$

Waarbij:

$$X_{ij} = \begin{cases} 1 & \text{als object j in knapzak i zit;} \\ 0 & \text{anders.} \end{cases}$$

We hebben nu een aantal standaard knapzak problemen beschreven. Deze problemen zijn echter niet genoeg om het slitproces bij Royal Smit te beschrijven. In de volgende twee paragrafen zullen we een tweetal knapzak problemen beschrijven die het slit proces bij Royal Smit beter zullen beschrijven.

2-dimensionaal knapzakprobleem

In het eerste geval gaan we ervan uit dat er niet voldoende moederrollen zijn om alle orderbreedtes uit te slitten. Het probleem lijkt nu al heel erg op het knapzak multiple-choice multiple knapzakprobleem probleem, behalve dat we orderbreedtes ook nog kunnen opdelen. We kunnen het probleem nu als volgt omschrijven:

Wat hebben we?

- We hebben een voorraad van m moederrollen $M_1, ..., M_m$.
- Elke moederrol M_i heeft een breedte $B(M_i)$ en een gewicht $G(M_i)$.
- We hebben een behoefte van n orderbreedtes $O_1, ..., O_n$.
- Elke orderbreedte O_j heeft een breedte $B(O_j)$ en een gewicht $G(O_j)$.

Wat willen we?

- We willen dat er zoveel mogelijk gewicht geslit wordt uit de voorraad moederrollen.

- Elke orderbreedte mag verdeeld worden in s stukken van dezelfde breedte die alle minder lang zijn en alle uit een moederrol geslit kunnen worden. we geven een partitie van een orderbreedte O_j in q delen aan met $P_q(O_j)$. Verder geven we het k-de deel van een partitie van O_j aan met O_{jk}.

We kunnen s afschatten door te kijken hoe vaak de langste orderbreedte in de kleinste moederrol past. We krijgen nu het volgende knapzakprobleem:

$$\max\{\sum_{i=1}^{n}\sum_{q=1}^{s} P_q(O_j)\sum_{k=1}^{q}\sum_{j=1}^{n} \qquad O_{ikj}G(O_{jk})|\sum_{j=1}^{n} G(O_{jk})O_{ikj} \leq G(M_i)\ \forall i = (1,.....,m)$$
$$\text{en } \forall j\ \sum_{i}\sum_{k} O_{ikj} \leq 1\}$$

Waarbij:

$$O_{ikj} = \begin{cases} 1 & \text{als deel k van orderbreedte j in moederrol i zit;} \\ 0 & \text{anders.} \end{cases}$$

We merken op dat het probleem steeds complexer wordt als we meer voorwaarden toevoegen. In het tweede geval gaan we ervan uit dat er wel voldoende moederrollen zijn om alle orderbreedtes uit te slitten. Het probleem is nu nog anders omdat we niet alleen het gewicht wat we willen slitten willen maximaliseren. We willen nu juist het afval minimaliseren zodanig dat alle orderbreedtes geslit zijn. We zien nu dat het slitprocess niet direct vertaald kan worden naar een knapzakprobleem. De knapzakproblemen die we in dit hoofdstuk en de relatie met het slitproces zijn wel een hele bruikbare tool als we het slitprobleem willen oplossen met behulp van lineair programmeren. Om een beter beeld te krijgen van het knapzakprobleem zullen we in de volgende paragraaf een oplossingsmethode bekijken.

oplossingen knapzak probleem

Voor het oplossen van het $[0,1]$ knapzak problemen is er een standaard methodes. De dynamische methode.

dynamische methode
De dynamische methode is erop gebaseerd om het probleem steeds te reduceren en zal het probleem recursief oplossen. We hebben voor deze methode de volgende functie nodig:
$F(m,s) = max\{\sum_{j=1}^{m} P_jX_j|\sum_{j=1}^{m} W_jX_j \leq s;\ X_i \in 0,1;\ \forall s \in (0,1,....,c);\ \forall m \in (,1,....,n)\}$
Deze functie geeft de maximale oplossing als we alleen objecten $1,2,...,m$ gebruiken en maximaal gewicht van de knapzak gelijk is aan s. Wat we dus willen oplossen is het probleem $F(n,c)$. Want dit geeft ons:
$F(n,c) = max\{\sum_{j=1}^{n} P_jX_j|\sum_{j=1}^{n} W_jX_j \leq c\}$ waar we naar opzoek waren. Hoe kunnen we dit nu oplossen? We kijken naar de recursie op $F(m,s)$. Allereerst merken we op dat

$F(m,0)$ gelijk is aan 0 want we kunnen de knapzak in dit geval niet vullen en dus is de maximale waarde van de knapzak met capaciteit gelijk aan 0. Verder geldt:

$$F(1,s) = \begin{cases} P_1 & \text{als } W_1 \leq s \leq c \\ 0 & \text{als } 0 \leq s \leq W_1 - 1. \end{cases}$$

Want $F(1,s) = max\{\sum_{j=1}^{1} P_j X_j | \sum_{j=1}^{1} W_j X_j \leq s\}$
$= P_1 X_1 | W_1 X_1 \leq s$

$$= \begin{cases} P_1 & \text{als } W_1 \leq s \leq c \\ 0 & \text{als } 0 \leq s \leq W_1 - 1. \end{cases}$$

Want als $W_1 \leq s$ dan is ook $W_1 X_1 \leq s$ omdat $X_1 \in \{0,1\}$. En als $W_1 \geq s$ dan is of $W_1 X_1 \geq s$ of $X_1 = 0$ en dus $P_1 X_1 = 0$.

Nu kijken we naar de recursie op $F(m,s)$. Er geldt:

$$F(m,s) = \begin{cases} F(m-1,s) & \text{als } 0 \leq s \leq W_m - 1 \\ maxF(m-1,s), P_m + F(m-1, s-W_m) & \text{als } W_m \leq s \leq c. \end{cases}$$

We merken op dat voor het berekenen van $F(m,s)$ we twee opties hebben: optie 1 laat object $m-1$ buiten beschouwing en dan krijgen we dus dezelfde oplossing als $F(m,s)$ want $F(m,s)$ was gedefinieerd als de beste oplossing om de $m-1$ objecten in een knapzak te stoppen met maximale capaciteit s.optie 2 nemen we wel object m mee in de knapzak (merk op dat dit alleen kan als het gewicht van object m kleiner is dan s want s is de maximale capaciteit van de knapzak. We krijgen nu voor $F(m,s)$ dat deze gelijk is aan $F(m-1,s)$ als er geen beter oplossing vbestaat die object m meeneemt, en we krijgen $P_m + F(m-1, s-W_m)$ als er wel een betere oplossing is met object m. We kunnen nu dus $F(n,c)$ berekenen door gebruik te maken van deze recursie.
voorbeeld dynamische methode We kijken nu naar een voorbeeld dat we zullen oplossen met de dynamische methode. We kijken naar het volgende $[0,1]$knapzak probleem. We hebben de volgende 4 objecten:
Object 1: gewicht 10 en waarde 12
Object 2: gewicht 4 en waarde 5
Object 3: gewicht 8 en waarde 10
Object 4: gewicht 3 en waarde 4
En we hebben een knapzak met capaciteit 17. Hoe kunnen we nu de grootste waarde meenemen in onze knapzak? Wiskundig beschouwen we het volgende probleem:
$max\{12X_1 + 5X_2 + 10X_3 + 4X_4 | 10X_1 + 4X_2 + 8X_3 + 3X_4 \leq 17\}$ in onderstaande tabel staat nu het resultaat van de berekeningen.
En we zien dat de maximale waarde van de knapzak die we mee kunnen nemen

21 is. We nemen dan mee: X_1,X_2 en X_4.
Zoals we in dit voorbeeld eenvoudig kunnen zien is het knapzak probleem voor grotere problemen niet eenvoudig op te lossen. Het oplossen van elk standaard knapzak probleem is een NP-hard probleem. Met behulp van integer programming en liniear programmeren denken wij wel dat er in vele gevallen een hele goede benadering kan worden gevonden van de optimale oplossing. Hiermee zou Royal Smit natuurlijk al heel tevreden zijn.

4.4 Lineair programmeren

Bij ons onderzoek naar slitproblemen kwamen we erachter dat er een duidelijke standaard in de industrie bestaat voor het oplossen van slitproblemen: Lineair programmeren.

Wat is Lineair Programmeren?

Lineair programmeren is een methode voor het vinden van een optimale (minimale of maximale) waarde van een wiskundig systeem, waarbij alle randvoorwaarden lineaire vergelijkingen zijn.
Omdat de functie die gemaximaliseerd moet worden (respectievelijk geminimaliseerd) lineair is, kan het maximum (resp. minimum) alleen maar op een rand van het domein liggen. Beter nog, de rand bestaat uit segmenten gevormd door de individuele lineaire ongelijkheden uit de randvoorwaarden, en binnen zo'n segment ligt het maximum (resp. minimum) weer op een uiteinde van het segment. Het gevolg is dat het maximum (resp. minimum) altijd op een knooppunt van twee (of meer) lineaire vergelijkingen ligt. Voor het vinden van het maximum (resp. minimum) hoeven we dus alleen de knooppunten bij langs te gaan.

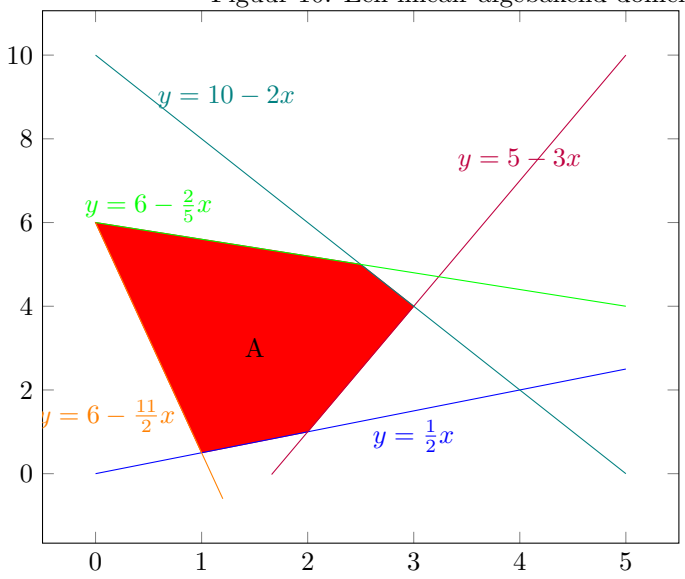
Lineair Programmeren en slitproblemen

Slitproblemen blijken geschikt te modelleren als systemen geschikt voor lineair programmeren. We hebben in ons onderzoek steeds weer gevonden dat lineair programmeren gebruikt wordt om slitproblemen op te lossen. Ook van Royal Smit hebben we te horen gekregen dat Lineair Programmeren een bekende term is in de industrie.

Een eenvoudig slitprobleem is het volgende: Een bedrijf heeft een (praktisch onbeperkte) voorraad moederrollen. De moederrollen hebben een standaardbreedte B_M en een standaard gewicht/lengte. Het bedrijf heeft orders voor smallere stroken metaal, met breedtes $\{B_1, B_2, B_3, ...B_n\}$, een bepaalde breedte B_i hebben ze O_i keren nodig, wat betekend dat ze O_i keer de volledige lengte van een moederrol aan materiaal moeten slitten. Het bedrijf wil restafval minimaliseren. Dit is te modelleren door naar de mogelijke slitpatronen te kijken, en slitpatronen zo te kiezen dat het minimaal restafval oplevert. Als model: Minimaliseer $\sum_j X_j R_j$ waarbij we j verschillende slitpatronen beschouwen. X_j is het aantal keer dat we patroon j gebruiken, R_j is hoeveel rest patroon j

s	$F(1,s)$	X_1	$F(2,s)$	X_2	$F(3,s)$	X_3	$F(4,s)$	X_4
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	4	1
4	0	0	5	1	5	0	5	0
5	0	0	5	1	5	0	5	0
6	0	0	5	1	5	0	5	0
7	0	0	5	1	5	0	5	0
8	0	0	5	1	10	1	10	0
9	0	0	5	1	10	1	10	0
10	12	1	12	0	12	0	12	0
11	12	1	12	0	12	1	14	1
12	12	1	12	0	15	1	15	0
13	12	1	12	0	15	1	16	1
14	12	1	17	1	17	0	16	1
15	12	1	17	1	17	0	19	1
16	12	1	17	1	17	0	19	1
17	12	1	17	1	17	0	21	1

Figuur 10: Een lineair afgebakend domein



oplevert. Er moet verder gelden dat: $\sum A_{i,j}X_j \geq O_i \forall i$ waarbij $A_{i,j}$ staat voor het aantal keer dat breedte B_i in patroon j voorkomt, want er moet genoeg van elke order geslit worden. Verder is belangrijk dat de slitpatronen geldig zijn, en dus niet te veel uit een rol proberen te slitten: $\forall j : \sum_i A_{i,j}B_i \leq B_M$. Ook moet gelden $\forall j : \mathbb{Z} \ni X_j \geq 0$ (i.e. X_j geheeltallig en positief) en net zo voor $A_{i,j}$.

Dit geeft een model van het probleem dat in theorie redelijk eenvoudig op te lossen valt. Het probleem is echter dat de oplossingsruimte veel te groot is. We kijken naar de verschillende slitpatronen die mogelijk zijn. Dit zijn er echter te veel om met computers snel een oplossing te vinden als we ze allemaal beschouwen.

Om toch snel oplossingen te vinden zijn er algoritmes die beginnen met minder slitpatronen en dynamisch de oplossingsruimte vergroten om een oplossing te vinden. Dit is de zogenoemde delayed column generation methode. Deze methode gebruikt ook het eerder genoemde knapzakprobleem om geschikte kandidaten te vinden om toe te voegen aan de met lineair programmeren op te lossen matrix.

De situatie bij Royal Smit

De beschrijving hierboven geeft een beeld van hoe een eenvoudig slitprobleem uitgedrukt kan worden als een lineaire functie met lineaire randvoorwaarden. Het slitprobleem bij Royal Smit is een stuk complexer, want de moederrollen kunnen in zowel breedte als lengte variëren. Een soort tussenvorm verkrijgen we door de lengte-restrictie op de moederrollen te negeren, maar wel verschillende breedtes mee te nemen. Het is handig om weer naar de verschillende mogelijke slitpatronen te kijken. Hierbij moet rekening gehouden worden dat de moederrollen verschillende breedtes hebben, en er dus ook verschillende slitpatronen geschikt zijn voor die moederrollen.

We sorteren de moederrollen op breedte, omdat voor moederrollen met gelijke breedte dezelfde slitpatronen geldig zijn. We beschouwen dan voor elke mogelijke breedte de slitpatronen op vergelijkbare manier als in het eenvoudige geval.

De te optimaliseren functie wordt dan:

$$min \sum_k \sum_j X_{j,k}R_{j,k}$$

Hier is $X_{j,k}$ hoeveel slitpatroon j in breedte k gebruikt wordt, uitgedrukt in lengte, en $R_{j,k}$ is hoeveel rest dat patroon in de breedte heeft per lengte-eenheid. Dit is dus een minimalisatie over de totale hoeveelheid rest in de breedte. De randvoorwaarden zijn dan:

$$\forall k \forall j \sum_i A_{i,j,k}B_i \leq M_k$$

waarbij $A_{i,j,k}$ het aantal keer is dat breedte B_i in slitpatroon j,k zit, en M_k de breedte is van de moederrol. En ook moet gelden:

$$\forall i \sum_k \sum_j A_{i,j,k} X_{j,k} B_i \geq O_i$$

Hier is O_i het totaal benodigde gewicht van orderbreedte B_i.

Als we wel de variabele lengte van de moederrollen meenemen wordt het probleem nog iets groter. Omdat alle moederrollen nu in zowel lengte als breedte kunnen verschillen, moeten op als we naar slitpatronen kijken de slitpatronen nu per moederrol bekeken worden. Omdat er op deze manier gigantisch veel gevallen bekeken moeten worden is het interessant om nog verder onderzoek te doen naar mogelijke andere modellen van het probleem.

Advies voor volgend jaar

De beschrijving hierboven is slechts een gedeeltelijke beschrijving van het probleem. Lineair programmeren was voor ons compleet nieuwe stof die we dus nog niet volledig beheersen, hierdoor is het moeilijk om een echt goede beschrijving van het probleem te geven. We denken echter wel dat het een veelbelovende methode is om dit probleem op te lossen, en als er volgend jaar weer een groepje studenten aan dit probleem werkt is dit zeker een mogelijkheid als ze vanaf het begin zich verdiepen in lineair programmeren.

5 Meswissels minimaliseren

In dit hoofdstuk zullen we een ander optimalisatie probleem bekijken. We kijken in dit hoofdstuk naar het minimaliseren van het aantal meswissels tijdens het slit proces. We zullen eerst de situatie bij Royal Smit beschrijven. Vervolgens zullen we dit vertalen naar een wiskundig model en aan de hand van dit model zullen we een aantal uitspraken doen die kunnen helpen bij het minimaliseren van de meswissels.

Hoe zit het bij Royal Smit

Bij Royal Smit hebben ze de volgende situatie. Er zijn per order een aantal moederrollen die geslit moeten worden volgens een gegeven slitoplossing. Deze moederrollen moeten één voor één door de slit machine. Omdat iedere moederrol eigen slitbreedtes heeft moeten de messen van de slit-machine tussen iedere moederrol opnieuw ingesteld worden. Het opbouwen van de messen gebeurd vanaf links. Het meest linker mes wordt eerst geplaatst en vervolgens wordt steeds het mes rechts ernaast afgesteld. Zo wordt de slitmachine vanaf de linker kant opgebouwd. Als de slitmachine tussen twee moederrollen opnieuw ingesteld moet worden gebeurd dit dus vanaf de rechterkant. De messen worden nu vanaf de rechterkant eerst los gemaakt en vervolgens weer vanaf links opgebouwd. Als

we goed naar dit proces kijken zien we dat voor twee moederrollen die na elkaar geslit worden de messen die aan de rechterkant op dezelfde plek zitten niet vervangen hoeven te worden, maar zodra het eerste mes aan de rechterkant vervangen moet worden weten we dat alle messen die rechts hiervan zitten opnieuw ingesteld moeten worden.
Het grote probleem bij het wisselen van de standen van de messen is dat dit zeer nauwkeurig moet gebeuren, daarom is het vervangen van de messen een tijdrovend onderdeel van het slitproces, het wisselen van één mes kost ongeveer 10 minuten. De kosten die Royal Smit hierbij maakt zijn niet alleen de manuren die nodig zijn voor het wisselen van de messen maar ook de tijd dat de machine niet kan draaien. Het niet laten draaien van de machine kan ook bovendien een vertraging van het productieproces opleveren. Reden genoeg om beter te kijken naar dit optimalisatie proces.
In de rest van dit hoofdstuk zullen we laten zien hoe Royal Smit tijd kan besparen in het slitproces. We zullen eerst het huidige systeem van Royal Smit beschrijven en aan de hand van een aantal voorbeelden laten zien dat er een efficiëntere manier is om het aantal meswissels te verminderen. Op dit moment wordt, afgezien van de rand, per moederrol de breedste orderbreedte links geplaatst en worden dan van links naar rechts de orderbreedtes op aflopende volgorde van breedte ingedeeld. Bijvoorbeeld, 5|800|130|5, of 5|600|220|130|5. Vervolgens worden de moederrollen met de breedste orderbreedte als eerste na elkaar geslit. Daarna nemen ze de moederrollen die over zijn met een breedste orderbreedte en zo gaat het proces door tot alle moederrollen van een order geslit zijn. Voor de volgende order ziet dit er dus als volgt uit:

Een voorbeeld als illustratie

Stel we hebben de volgende order bestaande uit tien moederrollen. Figuur 11 is een voorbeeld van de huidige slitmethode van Royal Smit. Elke regel staat voor een moederrol. En de getallen in een regel zijn de orderbreedtes die uit deze moederrol geslit worden. Verder bevat de afbeelding verticale lijnen. Deze staan voor een meswissel. Omdat bijvoorbeeld de linker rand bij alle tien moederrollen 5 is moet, het meest linker mes maar een keer ingesteld worden en hoeft verder niet veranderd te worden.

Als we het aantal meswissels bij deze order berekenen zien we dat er bij het slitten van deze order 24 messen verwisseld moeten worden. We zullen later in dit hoofdstuk zien dat we voor deze order het aantal meswissels kunnen reduceren tot ..... Maar we zullen nu eerst laten zien dat dit probleem niet eenvoudig is en lastiger is dan het op het eerste oog misschien lijkt.

Eerste idee:
Een eerste idee lijkt om de orderbreedte die in de meeste moederrollen voorkomt het meeste links te plaatsen. Dit klinkt aannemelijk maar met een eenvoudig voorbeeld in figuur 12 zien we direct in dat deze aanpak niet zo'n goed idee is.
Tweede idee:
Omdat het geen goed idee bleek om te kijken naar welke orderbreedtes hori-

5	900	130	5	
5	880	130	5	
5	860	140	5	
5	700	170	130	5
5	700	140	140	5
5	680	140	140	5
5	680	170	130	5
5	660	170	130	5
5	660	140	140	5
5	400	400	130	5

25 messen

Figuur 11: Voorbeeld slitoplossing met meswissels zoals op dit moment bij Royal Smit gebruikelijk

5	**A**	P	Q	5
5	**A**	S	T	5
5	**A**	**C**	**D**	5
5	**B**	**C**	**D**	5

11 messen

5	**C**	**D**	**A**	5
5	**C**	**D**	**B**	5
5	**A**	P	Q	5
5	**A**	S	T	5

10 messen

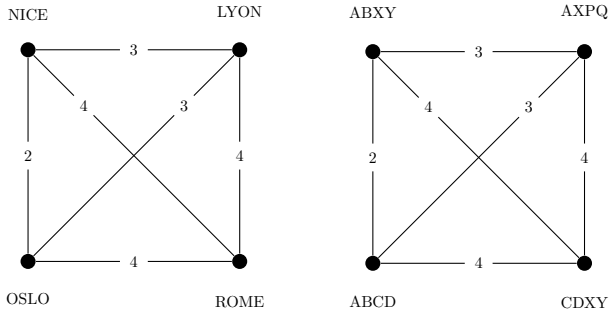
Figuur 12:

zontaal het meeste voorkomen hebben we als tweede idee gekeken naar welke combinatie orderbreedtes het vaakst voorkomt. En om de moederrollen die de meeste gemeenschappelijke orderbreedtes hebben na elkaar te slitten. Ook dit idee blijkt niet altijd de optimale oplossing te geven. Zo zien we bijvoorbeeld in figuur 12 dat er twee moederrollen zijn die drie orderbreedtes gemeenschappelijk hebben maar dat het toch efficiënter is om deze moederrollen niet na elkaar te slitten.

Derde idee:
Omdat de eerste twee ideeën in vele gevallen niet de optimale oplossing geven hebben we gekeken of het mogelijk is om via een brute force algoritme na te gaan wat de meest efficiënte oplossing is. We merken op dat het niet eenvoudig is om van een oplossing te zeggen of de oplossing optimaal is of niet. In vele gevallen zal de optimale oplossing ook niet uniek zijn. Voor een brute-force method kijken we naar een enigszins realistisch voorbeeld. Stel we hebben een order die we kunnen slitten uit dertig moederrollen en elke moederrol heeft gemiddeld drie orderbreedtes (inclusief offcut). Voor een brute force algoritme moeten we in dit geval van $30! * 3!^{30} \approx 10^{55}$ mogelijkheden langs gaan hoeveel meswissels er zijn. Er zijn namelijk 30! volgordes van moederrollen en voor de moederrollen zijn er gemiddeld 3! verschillende volgordes van slitbreedtes (exclusief rand). Een computer die 10^{12} van zulke berekeningen per seconde kan uitvoeren, doet daar zo'n 10^{30} jaar over! We zien nu dat het niet mogelijk is om een brute-force algoritme te ontwikkelen die binnen redelijke tijd de oplossing geeft. We begonnen steeds beter in te zien dat het probleem wellicht veel lastiger zou zijn dan in eerst instantie voorzien. Het probleem blijkt zelfs NP-hard te zijn. Om dit in te zien beschouwen we het handelsreizigersprobleem en merken we op dat ons probleem 'moeilijker' is dan dit NP-hard probleem. Betekent dit dat we geen beter algoritme dan brute-force kunnen vinden? Dat het probleem NP-hard is betekent niet dat er geen efficiënter algoritme is dan brute force. Voor het algemene geval is het probleem NP-hard maar we zullen laten zien dat we ons probleem kunnen reduceren tot een probleem dat in de meeste gevallen voor de computer binnen redelijke tijd oplosbaar is.

5.1 Handelsreizigersprobleem

Het Handelsreizigersprobleem is een bekend wiskundig optimalisatie probleem. Het probleem luidt: Wat is voor een handelsreiziger de snelste manier om elke stad, waar hij handel drijft, precies één keer aan te doen? We zullen nu de equivalentie tussen het Handelsreizigersprobleem en een eenvoudiger geval van het meswisselprobleem aantonen. We nemen het meswisselprobleem met het gegeven dat de volgorde van slitbreedtes in een moederrol vast ligt. We kunnen nu voor beide problemen een graaf maken die tussen elke tweetal knopen (steden en moederrollen) de tijd (in respectievelijk uren en meswissels) geeft. In het handelsreizigersprobleem zijn we op zoek naar de snelste route die alle steden aandoet en in het meswisselprobleem zijn we opzoek naar het pad dat de minste meswissels nodig heeft en alle moederrollen afgaat. Het is nu niet meer moeilijk



(a) Handelsreizigersprobleem

(b) Meswisselprobleem

Figuur 13: Handelsreizigersprobleem

om in te zien dat deze twee vragen equivalent zijn. Zie bijvoorbeeld figuur 13 voor een handelsreizigersprobleem en een equivalent meswissel probleem. In het oorspronkelijke Handelsreizigersprobleem was de vraag wat de snelste route was om alle steden aan te doen en weer te eindigen in dezelfde stad als waar de handelaar zijn handelsreis begint. Maar het blijkt dat het Handelsreizigersprobleem in beide gevallen NP-hard is. Omdat een makkelijkere variant van het meswisselprobleem equivalent is aan het Handelsreizigersprobleem weten we dat het meswisselprobleem een NP-hard probleem is. In de volgende paragraaf zullen we een algoritme laten zien die in vele gevallen van het meswisselprobleem bij Royal Smit de optimale oplossing zal geven, hiervoor zullen we een wiskundig model maken.

5.2 Wiskundig model meswisselprobleem

- Omdat de machine de messen van links opbouwt kunnen we orders die niet dezelfde rand hebben los van elkaar beschouwen.
- Als groepen moederrollen disjunct zijn kunnen we deze groepen apart beschouwen.
- Als een breedte maar in een moederrol voorkomt kunnen we deze standaard het meest rechts zetten omdat deze breedte nooit een mes wissel kan verminderen.
- Als een breedte in een moederrol strikt meer voorkomt dan in alle andere moederrollen dan kunnen we deze breedte net zo vaak het meest rechts zetten tot we alleen het aantal van deze breedtes overhouden wat wel in minstens één andere moederrol voorkomt.

We zullen nu een wiskundig model maken van het meswisselprobleem waarbij we bovenstaande aannames mee zullen nemen.

Welke trivialiteiten mogen we aannemen

Voor dit optimalisatie proces zijn er een aantal eigenschappen die voor het vinden van een goede oplossing van belang kunnen zijn. We houden voor alsnog buiten beschouwing dat randen variabel zijn en dat we linker en rechter rand kunnen omwisselen. Hier komen we later op terug.

- (i) We kunnen de moederrollen met verschillende breedtes van de linkerrand onafhankelijk van elkaar beschouwen, omdat bij moederrollen met verschillende randen alle messen opnieuw ingesteld moeten worden. We delen nu de moederrollen op in groepen. Elke moederrol wordt toegevoegd aan de groep met moederrollen die dezelfde rand hebben aan de linkerkant.

- (iia) Twee moederrollen in dezelfde groep met identieke slitbreedtes zullen we altijd na elkaar slitten en de slitbreedtes willen we in beide moederrollen in dezelfde volgorde hebben. Het is namelijk in geen enkel geval efficiënter om twee dezelfde moederrollen niet na elkaar te slitten. Het identiek na elkaar slitten van de moederrollen kost namelijk geen meswissel.

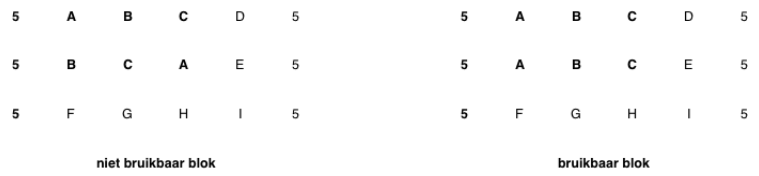
- (iib) De meest rechter rand mogen we buiten beschouwing laten want als twee moederrollen binnen een groep dezelfde orderbreedtes hebben dan slitten we deze na elkaar en als de moederrollen niet dezelfde orderbreedtes hebben moet er minstens een mes opnieuw ingesteld worden en dus moet ook het meest rechter opnieuw ingesteld worden.

- (iiia) Een slitbreedte die slechts één keer in een groep voorkomt mag zonder verlies van algemeenheid als meest rechter breedte geslit worden.

- (iiib) Een slitbreedte die in een order vaker voorkomt dan in de andere moederrollen uit dezelfde groep mag net zo vaak als meest rechterbreedte geplaatst worden tot we evenveel van deze breedtes overhouden als er in minstens één andere moederrol van deze groep zitten.

- (iv) Als een groep moederrollen uiteenvalt in disjuncte verzamelingen, dan mogen we de disjuncte verzamelingen onafhankelijk van elkaar beschouwen. Hier bedoelen we met disjunct dat de verzameling slitbreedtes disjunct is. We delen elke groep op in disjuncte sub-groepen zodanig dat de sub-groepen binnen een groep disjunct zijn.

Om dit te vertalen naar een wiskundig probleem krijgen we nu het volgende. We hebben een verzameling moederrollen $\{M_1, M_2, ..., M_n\}$ en voor elke moederrol M_i hebben we bijbehorende breedtes die uit deze moederrol gesneden worden. We noemen deze $M_{i,1}, M_{i,2}, .., M_{i,k}$. Waarbij we krijgen dat $M_{i,1}$ de standaard rand links is van moederrol M_i en $M_{i,k}$ de breedte van de rechter rand van moederrol M_i is. We delen nu eerst de moederrollen op in groepen



Figuur 14:

met verschillende randen links. We delen vervolgens elke groep op in disjuncte sub-groepen, zoals hierboven beschreven. We hoeven in elke sub-groep alleen te kijken naar de slitbreedtes, want de linker rand is overal hetzelfde en de rechter mogen we negeren volgens (iib). Omdat de meeste van onze moederrollen tussen de twee en zes slitbreedtes (inclusief rand) hebben hoeven we in elke sub-groep alleen maar te kijken de slitbreedtes, waarvan er in bijna alle gevallen tussen de twee en vier zijn.

Definitie 1 *We noemen een blok een combinatie van slitbreedtes die in minstens twee moederrollen uit dezelfde groep voorkomt. We zeggen dat een blok lengte n heeft als er n moederrollen in een groep zijn die deze combinatie slitbreedtes bevat en we zeggen dat een blok breedte m heeft als het blok een combinatie van m slitbreedtes is. Een blok van lengte n en breedte m noemen we een n-bij-m blok.*

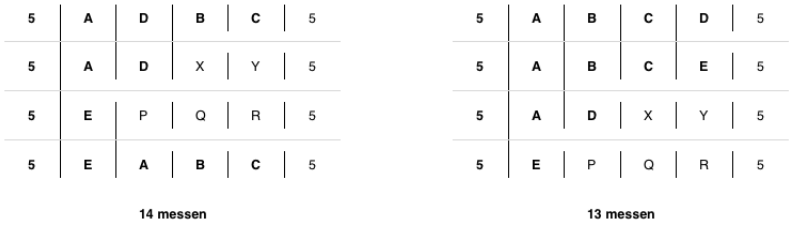
Definitie 2 *We zeggen dat een blok bruikbaar is als de volgorde van de slitbreedtes uit het blok in alle moederrollen uit het blok gelijk is.*

Lemma 1 *Als in een sub-groep elke moederrol maximaal vier slitbreedtes heeft en er is precies één blok van breedte drie in het blok, dan willen we alle moederrollen met deze combinatie slitbreedtes als een bruikbaar blok na elkaar slitten en het blok dat op deze manier verkregen wordt het meest links zetten.*

Bewijs 1 *Stel we hebben een* 2*-bij-*3 *blok en dit is het enige blok van breedte drie. We mogen zonder verlies van algemeenheid aannemen dat de twee moederrollen uit dit blok niet gelijk zijn, als dit wel het geval is willen we vanwege (iia) de moederrollen altijd elkaar slitten en de slitbreedtes in dezelfde volgorde hebben.*

- *Als beide moederrollen, M_1 en M_2, precies drie slitbreedtes hebben zijn we klaar want dan willen we deze moederrollen zeker na elkaar slitten.*
- *Stel precies één van de moederrollen heeft vier slitbreedtes (excl. rand) heeft (zeg M_2). Het bruikbaar na elkaar slitten betekent in dit geval dat de 3 slitbreedtes uit het blok in M_2 links worden gezet, want het blok staat*

Figuur 15:

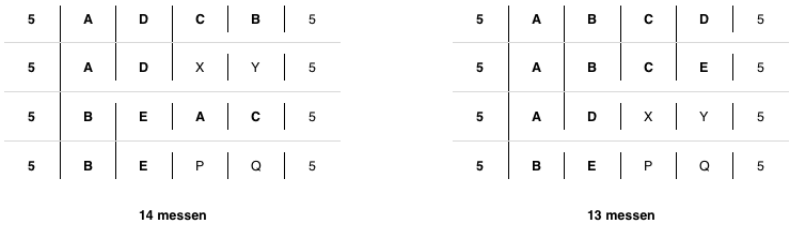


Figuur 16:

in M_1 op triviale wijze al links. Dan weten we dat de vierde slitbreedte in deze moederrol minstens een keer in een andere moederrol zit want anders mogen we deze slitbreedte zeker als meest rechter slitbreedte nemen (iiia). Elke moederrol M_i ongelijk M_2 met deze slitbreedte heeft maximaal een 2-tupel gemeenschappelijk met de moederrol M_1 die maar drie slitbreedte heeft, anders hebben we een nieuwe 3-tupel van orde twee. De vierde slitbreedte zit dus in geen enkel 3-blok. Het is voor deze moederrol dus altijd het efficiëntste om de moederrol in het 3-blok te slitten.

- *Stel beide moederrollen hebben vier slitbreedtes. We mogen zonder verlies van algemeenheid aannemen dat de vierde slitbreedte van beide moederrollen ongelijk zijn en ook dat beide slitbreedtes minstens in een andere moederrol voorkomen. Als beide orderbreedtes niet in een 2-blok zitten maken we duidelijk geen winst door deze als linker breedte te slitten (Figuur 15). In het slechtste geval zitten beide orderbreedtes, die niet in het 3-blok zitten, wel in een 2-blok. Maar ook in dit geval is het efficiënter om het 3-blok links te slitten (Figuur 16). Dit geldt ook in het algemeen, omdat voor een 3-blok van lengte k het weghalen van de laatste moederrol altijd minstens één meswissel meer kost.*

Stelling 1 *Als voor een 3-blok geldt dat alle moederrollen in dit 3-blok maar in*



Figuur 17:

één drieblok zitten in de sub-groep dan willen we dit 3*-blok als bruikbaar blok slitten.*

Deze stelling is een gevolg van het eerdergenoemde lemma. tussen elke moederrol hoeven we namelijk maar maximaal één mes te wisselen. Het niet gebruiken van een moederrol in dit blok levert altijd minstens één meswissel meer op. Dit is onafhankelijk van de andere moederrollen in de groep.

We kunnen hiermee het meswisselprobleem van Royal Smit heel erg verkleinen. We merken wel op dat de over volgorde van het 3-blok niks kunnen zeggen. Een laatste opmerking die we maken is dat in een gemiddelde order van Royal Smit vaak veel moederrollen met twee slitbreedtes zitten. Dit zorgt ervoor dat er in vele gevallen meer sub-groepen zullen zijn. Als we over alle subgroepen een brute-force algoritme toepassen denken wij dat het meswisselprobleem van Royal Smit voor vele orders binnen redelijke tijd de optimale oplossing gevonden kan worden. als we naar meerdere orders kijken wordt het moeilijker om een algoritme te vinden die de optimale oplossing geeft.
Als we terug kijken naar ons voorbeeld in Figuur 11 kunnen we met de trivialiteiten alleen al een oplossing vinden die slechts 15 meswissels heeft (zie Figuur 18). Ondanks we niet met zekerheid kunnen zeggen dat we de optimale oplossing altijd zullen vinden zullen we in vele gevallen een flink aantal meswissels kunnen verminderen. Hiermee hopen we Royal Smit te kunnen helpen met het winnen van tijd in het slitproces. Het zou heel mooi zijn als een volgend groepje aan dit probleem verder kan uitwerken en implementeren.

5	130	170	700	5
5	130	170	680	5
5	130	170	600	5
5	130	400	400	5
5	130	900	5	
5	130	880	5	
5	140	860	5	
5	140	140	700	5
5	140	140	680	5
5	140	140	660	5

16 messen

Figuur 18:

6 Inkoopadvies

Als het bedrijf de opdracht krijgt een nieuwe transformator te maken worden een aantal moederrollen besteld. Hierbij worden wat randvoorwaarden gegeven maar het is binnen een bepaald bereik willekeurig wat voor moederrollen het bedrijf daadwerkelijk krijgt. Nu is het de vraag of kosten bespaart kunnen worden door meer eisen op de inkoop te leggen.
Om dit na te gaan hadden we twee ideeën. Stel er is al een werkend programma dat voor een gegeven voorraad moederrollen met hoge garantie een goede oplossing geeft. Dan is het makkelijk om van de voorraad aan moederrollen een 'oneindige lijst' te maken. Dat kan door gewoon elk mogelijke moederrol zo vaak in de voorraad te zetten dat in principe elke orderbreedte uit elke moederrol geslit zou kunnen worden.
Een ander idee is om statistisch onderzoek te doen naar de orders die in het verleden geslit werden. Wij hebben het vermoeden dat bepaalde orderbreedtes vaak samen voorkomen. Als deze alvast perfect in moederrollen ingedeeld kunnen worden is minder materiaalverlies nodig.
Het implementeren en verder uit zoeken laten we over aan een volgend groepje.

7 Meerdere voortzettingen van het project

Bij het afsluiten van ons project kwam naar hoe voren hoe het volgende groepje het beste kan doorgaan. Uiteraard kan de situatie veranderd zijn. Ons advies slaat op een onveranderde situatie, en we stellen voor om onverdeelde aandacht te hebben voor één van deze:

- Zet al je aandacht in op lineair programmeren.
- Begin aan de code te schrijven en implementeer een boomdoorzoekend algoritme.

Maar als je met twee groepjes hieraan gaat werken is het implementeren en de algoritmiek te verdelen. Het zal ook heel waardevol voor je projectervaring zijn om zo samen te werken.

8 Tips voor het volgende groepje

- Vorig jaar had er precies maar één iemand van dat groepje aan de code gewerkt. Ze gaven ons het advies om met meer aan de code te werken en dat is naar onze mening ook veel beter.
- De vergaderhokjes in het studielandschap zijn ontzettend fijn om met je groepje in te werken

Referenties

[1] https://stackoverflow.com/questions/687731/breadth-first-vs-depth-first [05-07-2017]

[2] http://composingprograms.com [10-07-2017]

[3] https://en.wikipedia.org/wiki/Depth-first_search [11-07-2017]

[4] https://en.wikipedia.org/wiki/Breadth-first_search [11-07-2017]

9 Logboek

14-02-17

We hebben vandaag onze eerste bijeenkomst gehad. Vorig jaar heeft een groepje al een begin gemaakt aan het project van Royal Smit. Met z'n vieren hebben we het verslag van vorig jaar doorgelezen en besproken. Verder hebben we enkele vragen opgesteld die we vrijdag willen stellen. We hebben vrijdag de eerste afspraak met de begeleider van Royal Smit. Het allerbelangrijkste is dat we er vrijdag achter willen komen wat Royal Smit van ons verwacht. Hiervoor hebben we enkele vragen opgesteld zodat we een duidelijk beeld krijgen van wat er van ons verwacht wordt. Uit het verslag van vorig jaar kunnen we opmaken dat het belangrijk is om zelf initiatief te nemen en op tijd te beginnen.

17-02-17

Vandaag hebben we een rondleiding gehad bij Royal Smit. Mark heeft ons uitgelegd hoe een transformator gebouwd wordt en het hele proces uitgebreid laten zien. In het bijzonder liet hij ons zien hoe het slit proces bij Royal Smit in elkaar zit. Bij dit proces zullen wij proberen een bijdrage aan de efficiëntie te geven. We hebben een afspraak voor 27-02-17 gemaakt. We zullen dan daadwerkelijk het probleem bespreken waar wij ons tijdens het modellen practicum mee zullen bezig houden.

21-02-17

We zijn bij elkaar gekomen om vragen te bedenken en samen een beeld te vormen van de opdracht en het probleem. We willen graag na maandag echt aan de slag gaan en beginnen. We hebben enkele vragen via mail opgestuurd zodat Mark er al over na kan denken en op maandag een antwoord kan geven. Na de rondleiding van afgelopen vrijdag blijken er drie in verband staande optimaliseringsprocessen te zijn. Voor ons is het nog niet helemaal duidelijk wat de precieze vraagstelling is en wat er precies van ons verwacht wordt. We zien zelf op dit moment verschillende optimalisatie problemen en willen graag weten of er een verband is en wat er van ons verwacht wordt.

- Het optimaliseren van het inkoopproces
- Het optimaliseren van de slitoplossingen, welke orderbreedtes slitten we uit welke moederrollen?.
- Het optimaliseren van het slitproces, welke moederrollen slitten we eerst?
- Het optimaliseren van het doorverkopen van de resten

Op dit moment is het ons niet helemaal duidelijk waar onze hulp precies gevraagd wordt. Het lijkt ons handig om na maandag de gegevens (parameters) te hebben, die we voor het probleem nodig hebben om aan de slag te kunnen gaan. We zouden het fijn vinden als we op maandag van de volgende vragen diegene kunnen bespreken die voor ons relevant zullen zijn: Wat zijn de relevante parameters? (Breedtes, lengte, dikte, kostprijs, (inkoop en verkoop), tijd om een transformator te slitten enz..) Voor de inkoop van moederrollen is er een levertijd? Welke stukken staal kunnen jullie doorverkopen en voor welke prijs? Hoe zit het precies met de verschillende gewichten staalrollen? Wat zijn precies de verschillen ten opzichte van vorig jaar?

27-02-17

We zijn vandaag bij Royal Smit geweest voor een presentatie van Royal Smit over de probleemstelling en de verwachtingen voor het project. Het probleem is vanuit Royal Smit opgedeelt in drie fases. Fase 1 is het creëren van een werkbare oplossing voor het slitten van orders uit gegeven rollen. Fase 2 gaat hier verder op door waarbij ook een inkoopadvies kan worden gegeven In Fase 3 wordt verder vooruit gekeken naar orders voor het lopende jaar, en geeft aan of het beter is zelf breedtes te slitten of ze in te kopen. Prioriteit ligt bij Fase 1, als dit werkend is kan er gekeken worden naar de volgende fases. Fase 1 is ook waar het vorige groepje mee bezig is geweest, en waar ze een gedeeltelijke oplossing voor gevonden hadden. De randvoorwaarden voor het probleem zijn echter iets veranderd en dus zal ook het programma veranderd moeten worden. Afspraken: -Bij het maken van het programma meer uitleg/commentaar schrijven. -Voor de volgende bijeenkomst ieder voor zich opschrijven hoe het probleem in elkaar zit, het verslag van vorig jaar doorlezen en het Cutting Stuck Problem doorlezen en kijken of het relevant is voor ons probleem. -Relevante artikelen in drive stoppen.

02-03-17

Vandaag hebben we met z'n vieren alles doorgenomen en een taakverdeling gemaakt voor de komende weken. We hebben de volgende dingen kort besproken: Voor fase 1 hebben we overwogen om eerst te proberen het programma van vorig jaar aan te passen en te kijken of we een programma kunnen maken dat aan de nieuwe eisen van Royal Smit voldoet. De verschillen waar we naar gaan kijken zijn: in het programma van vorig jaar gingen ze uit van meerdere orders

tegelijk. Uit ervaring blijkt dat dit voor Royal Smit niet efficiënt genoeg is om hiermee door te gaan en willen ze nu proberen om het slit process per order te bekijken. Omdat het slit proces een complex proces is heeft Royal Smit ervoor gekozen om niet meer alle verschillende materialen zelf te slitten maar zelf maar 2 materialen te slitten. Om minder verlies te maken wil Royal Smit gaan kijken naar het efficiënt doorverkopen van standaardmaten aan hun dochteronderneming in Nürnberg. Deze standaardbreedtes zijn: (130, 140, 150, 160, 170) Om het slit process te optimalizeren wil Royal Smit gaan kijken naar de winst die ze kunnen halen door het slit process te versnellen. Uit eigen ervaring maar ook uit onderzoek op de markt waarin Royal Smit zich bevindt heeft Royal Smit besloten om het slit process te versnellen door te proberen het aantal mes wisselingen te proberen te verminderen. minimale offcut willen ze op 50mm stellen. De slit breedte mag niet smaller zijn dan 50mm maar dit zal geen probleem zijn aangezien de minimale offcut ook al 50 mm moet zijn. aan beide kanten van de moederrol moet de offcut minimaal 5mm zijn. Als restant moederrol kleiner of gelijk is aan 40mm dan zoom standaard 2 x 20mm. een vraag die bij ons opkomt is of we dezelfde order breedtes binnen een order bij elkaar op mogen tellen of niet. Ook vragen we ons af of er bij de gevraagde order breedtes een marge is ingebouwd voor het eventuele verlies van de "20 meter" + " 3 omwentelingen". De 20 meter die gebruikt wordt als testmateriaal moeten we dit dan per moederrol eraf halen? of als we een moederrol halverwege stoppen moeten we dan opnieuw 20 meter weg gooien. Wat moet er gedaan worden en hoe gaan we dit aanpakken? Fase 1 moet aangepast worden: Voor het aanpassen van fase 1 zullen de volgende dingen gedaan/bekeken moeten worden, allereerst zullen we het programma van vorig jaar moeten begrijpen en kijken of we hier mee verder kunnen. Voor fase 1 gaan we ook kijken naar hoe de cod Verder gaan we kijken of het voor ons probleem een mogelijkheid is om naar een lineair programmeer probleem te vertalen.

05-03-17

Grzegorz en Rebekka hebben het programma van vorig jaar met hulp van Anita ingelezen en de eerste aanpassingen van de randvoorwaardes van fase 1 gedaan.

07-03-17

Afgelopen weekend hebben Gregorz en Rebekka veel tijd besteed aan het oude programma te begrijpen en zodanig aan te passen dat we met de nieuwe gegevens/eisen aan de slag kunnen gaan. Vandaag hebben we met samen gezeten om het nieuwe programma te bespreken en na te gaan wat er aangepast moet worden en op welke manier we dit het beste kunnen gaan doen. Dit willen we meenemen naar fase 2: In het oude programma zijn dezelfde order breedtes niet bij elkaar opgeteld, misschien is dit wel handig en efficiënt om naar te kijken. Als we gaan kijken naar de doorverkoop naar het zusterbedrijf zullen we het programma moeten aanpassen dat er eerst geoptimaliseerd wordt op een minimale offcut. we moeten nog nadenken over de slitbreedte tussen 40-50 mm.

(OOK IN FASE 1). Robin en Jorrit zullen deze week kijken naar het implementeren van de doorverkoop van standaardbreedtes aan een zusteronderneming en ook of het voor ons (in fase 2) een mogelijkheid is om te kijken naar lineair programmeren. We weten al dat dit mogelijk zou zijn als lengte en breedte van de moederrol niet variabel zijn maar willen toch gaan onderzoeken waarom dit voor ons wel/niet een alternatief is om naar te kijken. Grzegorz en Rebekka zullen deze week kijken naar de output van het programma. we gaan kijken naar het optimaliseren van zo min mogelijk mes wisselingen. Dit doen we in eerste instantie door de resultaten van het programma te sorteren en te kijken naar efficiëntie van messen wisselen. (OOK IN FASE 1). voor fase 2 moeten we kijken of het efficiënt is om dit mee te nemen in het process of dat het voor ons probleem makkelijker is om deze efficiency achteraf te berekenen. het percentage gebruikt voor order klopt niet. (OOK IN FASE 1). vragen die we aan Mark hebben gesteld zijn: Vorig jaar heeft het groepje gewerkt met een 10Wat houdt SO item in? We merken dat in de output van vorig jaar aan elke geslitte rol een a,b,c of d word toegevoegd. Deze zijn afhankelijk van het SO item? in hoeverre zijn offcuts tussen de 40 en 50mm toegestaan? of moeten deze helemaal vermeden worden?

09-03-17

We hebben het programma aangepast zodat het nooit oplossingen genereert met offcuts tussen de 30 en 50 mm. Ook hebben we het meeslitten van standaardbreedtes toegevoegd aan het programma. Dit hebben we gedaan in de functie slitten. Het programma gaat na het genereren van een oplossing zo veel mogelijk standaardbreedtes in het restant slitten. Dit telt niet als verlies mee. Alle gegenereerde oplossingen worden hierdoor efficiënter (minder verlies). Nieuwe code staat commentaar boven. vragen: is de terugverkoopprijs van offcuts nog steeds 0.8 en is de inkoopprijs 2.55?

14-03-17

Vandaag gaan we bespreken hoe we aan fase 2 kunnen beginnen. Verder gaan we fase 1 langzaam proberen af te ronden. We kunnen nog wel meer aanpassingen doen aan de code van fase 1 maar we denken dat we op dit moment beter aan fase 2 kunnen beginnen zodat we 5 april gericht vragen kunnen stellen over fase 2. Hoe gaan we verder/wat moet er nog gedaan worden: het knippen van moederrollen als deze niet volledig gebruikt worden (Fase 2) er moet één leverancier per order gebruikt worden. het berekenen van de totale efficiency het optimaliseren van de mes-standen het oplossen van het probleem dat het programma niet de hele order afmaakt maar sommige order breedtes niet afgemaakt worden. Het verslag van fase 1 voorbereiden presentatie fase 1

16-03-17

Jorrit en Robin hebben gewerkt aan het implementeren van één leverancier per order en het afmaken van alle order breedtes. Beide zijn gelukt, maar we hebben wel een nieuwe bug ontdekt: Als het programma een order breedte niet in een keer kan slitten, dan wordt die order gedeeltelijk geslit, waardoor het benodigde gewicht van de order afneemt. En daarmee ook het gpb van die order. Het programma hoort dan de lijst met orders opnieuw te sorteren op gpb, omdat het eerst de order met grootste gpd wil slitten. Maar de functie die daarvoor gebruikt werd werkt niet. Als we een wel werkende sorteerfunctie implementeren ontstaan er nieuwe bugs omdat het programma er later van uit gaat de ordening in de tussentijd niet veranderd is. Wat ons eerst een bug leek maar geen bug is, is dat in de output (het excel bestand) het gewicht van de orders in het algemeen afloopt, maar soms er opeens weer hoge gewichten onder staan. Dit komt omdat de lijst altijd gesorteerd is op het gpb van de moederrollen (van klein naar groot). Het algoritme kijkt wel eerst naar de grootste orders, en zoekt daar een zo'n klein mogelijke moederrol bij, waardoor vaak de kleinste moederrollen uit de oplossing de grootste order gewichten bevatten.

17-03-17

Grzegorz en Rebekka hebben gewerkt aan het implementeren van een totale efficiency en aan het implementeren van het sorteren van de messenstanden. De totale efficiency is toegevoegd. Het blijkt het meest efficiënt om de linker rand zo vaak mogelijk op dezelfde waarde te zetten. Vervolgens moet per gelijke linker rand gekeken worden hoe vaak een bepaalde breedte uit alle moederrollen geslit wordt. Hierbij wordt elke breedte per moederrol maar een keer geteld ook wordt er een breedte vaker uit een moederrol geslit. De moederrollen worden dan zodanig op volgorde gezet dat de gemeenschappelijke onderbreedten achter elkaar geslit worden. Dit te implementeren bleek echter moeilijk. We zijn begonnen te implementeren dat de gelijke linker randen bij elkaar staan. Maar ook dit is in het huidige programma moeilijk. Als we van plan zijn om voor fase 2 een nieuw programma te maken lijkt het ons het beste om daarbij (zoals bij object-orientatie) een onderscheid te maken tussen model, view en controller zodat heel erg duidelijk is in welke functies berekeningen gemaakt worden en welke functies alleen een outfile/cout geven wordt. Dat maakt het makkelijker om berekeningen toe te voegen. Opmerking voor volgende keer: de offcuts worden niet goed weergegeven als er ook een standaardbreedte geslit wordt en er het programma loopt vast als er voor geen leveranciers een oplossing is. Vragen voor fase 2: Wat is de levertijd van moederrollen Wat zijn de kosten voor het inkopen

24-03-17

05-04-17

Vandaag hebben we onze resultaten aan Royal Smit gepresenteerd. We hebben onze resultaten en bevindingen voor fase 1 gepresenteerd en onze vragen voor fase 2 opgesteld. Tijdens de presentatie hebben hebben we de volgende dingen besproken. onze resultaten van fase 1 onze vragen voor fase 2 wat willen we nu gaan doen hoe willen we dit gaan doen wat hebben we nodig van Royal Smit vragen/opmerkingen van Royal Smit

10-04-17

Wat gaan we nu concreet doen: FASE 1 AFMAKEN (VOOR 1 MEI) VARIABELE RANDEN INLEZEN BESTANDEN MES STANDEN SORTEREN HALVERWEGE AFKNIPPEN ZUSTER OFFCUTS 10c PER KILO TESTEN MEERDERE ORDERS FASE 2 LINEAIRE PROGRAMMEREN INLEZEN LINEAIRE PROGRAMMEREN NADENKEN OVER VERGELIJKINGEN EN RESTRICTIES FASE 1/2 ANALYSEREN PROGRAMMA TESTEN INVLOED VARIABELEN GEMIDDELDE EFFICIENCY In mei spreken we met Royal Smit om onze bevindingen en resultaten te bespreken.

18-04-17

vandaag hebben we besproken hoe we verder willen gaan met fase 2. Willen we een nieuw algoritme gaan schrijven met behulp van lineaire programmeren of niet? Gaan we het algoritme van fase 1 analyseren en waar mogelijk verbeteren? Wat gaan we nu concreet doen: FASE 1 AFMAKEN (VOOR 1 MEI) VARIABELE RANDEN V INLEZEN BESTANDEN MES STANDEN SORTEREN HALVERWEGE AFKNIPPEN ZUSTER OFFCUTS 10c PER KILO WINST VOOR SLITTEN Willen we dit meenemen in ons programma TESTEN MEERDERE ORDERS FASE 2 LINEAIRE PROGRAMMEREN INLEZEN LINEAIRE PROGRAMMEREN We hebben ons ingelezen en zijn er nog niet over uit hoe we nu verder gaan NADENKEN OVER VERGELIJKINGEN EN RESTRICTIES FASE 1/2 ANALYSEREN PROGRAMMA TESTEN INVLOED VARIABELEN GEMIDDELDE EFFICIENCY

20-04-17

We hebben vanmorgen met meneer Bosma besproken hoe we verder gaan met het project. We vinden het jammer dat we niet genoeg tijd hebben om lineair programmeren helemaal uit te werken, maar willen hier toch naar kijken. Ook gaan we het meswisselprobleem proberen beter te formuleren en op te lossen.

21-04-17 tot en met 26-04-17

Deze week hebben we in verschillende samenstellingen gekeken naar lineair programmeren. We hebben geprobeerd zo veel mogelijk erover te leren maar we zijn er achter gekomen dat het niet mogelijk is om binnen een paar dagen tijd lineair programmeren onder de knie te krijgen.

03-05-17

We hebben vanmorgen met meneer Bosma onze voortgang besproken. Lineair programmeren blijkt te moeilijk om te implementeren en de resterende tijd. Wat we wel graag willend doen is in het verslag duidelijk uitleggen hoe alles in elkaar zit. Dit blijkt namelijk soms niet eenvoudig.

04-05-17

Vandaag zijn we met zn vieren een hele dag bezig geweest om de volgende dingen te bespreken. Ook hebben we een sharelatex document aangemaakt zodat we allemaal tegelijkertijd aan het verslag kunnen werken.

09-05-17

Vandaag is Mark langs geweest omdat de voortgang te bespreken. We hebben hem uitgelegd dat het programma van vorig jaar niet meer heel 'çlean' is en het heel lastig is om dit programma als fundament te hebben voor een goed werkend programma. We willen dit in het verslag uitgebreid uitleggen en behandelen.

16-05-17

We hebben vandaag gekeken hoe we van de binary een executable maken opdat er geen tussenkomst is van codeblocks.

19-05-17

Tijdens het gesprek met Marco bleek dat ons programma minder goed was dan de oplossing van Marco. We konden niet meteen vinden waarom dit zo was. Hier willen we nog goed naar kijken.

23-05-17

Na het gesprek met Marco van afgelopen vrijdag zijn we vandaag weer samen gekomen om een en ander te bespreken. We zijn er helaas vrijdag achter gekomen dat Marco een slimmere oplossing heeft dan ons programma voor een bepaalde order. Het probleem was echter dat er maar weinig moederrollen te beschikking waren. Marco had handmatig een oplossing gevonden van 38 moederrollen en had nog 2.763 kilo niet ingedeeld. Ons programma gaf pas een volledige oplossing bij het toevoegen van een moederrol van 4.700 kilo. Een probleem waar

we pas vrijdag achter kwamen is dat ze vaak werken met minimale voorraden en zelfs vaker met voorraden die niet voldoende zijn om de hele order te slitten. Ons programma is hier niet op geprogrammeerd en zal in deze gevallen geen oplossingen geven. We gaan nu kijken hoe we dit beter kunnen aanpakken en willen dit in het verslag vermelden en waar mogelijk nog implementeren. We hebben de taken voor het verslag als volgt verdeeld.
Titelpagina (Rebekka)
Inleiding
probleem vaststellen/beschrijven (Grzegorz)
verschillende algoritmes (Rebekka)
lineair programmeren + knapzak probleem (Robin+Jorrit)
OO-stukje (Grzegorz)
meswissels (Jorrit)
programma uitleggen voor volgend groepje + Royal Smit
efficientie programma Voorbeelden: best/worst case (Robin)
ideeën voor volgende groep
tips voor volgende groep (iedereen)
logboek (Jorrit)
Referenties
Iedereen werkt komende week zelf aan het verslag en hebben afgesproken volgende week weer bij elkaar te komen.

09-06-17

Met z'n vieren hebben we vandaag de hele dag aan het verslag gewerkt, we hebben gekeken wat we al hadden en besproken welke delen al goed zijn, waar er nog aanpassingen nodig zijn en wie dit gaat doen en welke stukken nog geschreven moeten worden.

16-06-17

We hebben samen een planning en taakverdeling gemaakt voor het verslag en de presentatie. Rebakka en Grzegorz zullen presenteren en dus de presentatie maken, terwijl Jorrit en Robin meer aan het verslag zullen werken. We hebben allemaal aangegeven dat de tentamens eraan zitten te komen en dat we in de week van 3 juli afspreken om het verslag en de presentatie verder af te ronden. Met Royal Smit is afgesproken om op 10 juli de eindpresentatie te geven.

03-07-17 tot en met 07-07-17

Deze week heeft iedereen aan zijn deel van het verslag en de presentatie gewerkt. We hebben iedere dag eerst de voortgang besproken. Erna werkten we aan het verslag en de presentatie en aan het einde van de dag bespraken we weer de voortgang. Op vrijdag hebben we gekeken naar de presentatie en het verslag en besproken wat er nog gedaan moet worden. In het weekend zal iedereen zijn/haar stuk afmaken.

10-07-17

Vandaag hebben we de eindpresentatie aan Royal Smit geleverd. We hebben eerlijk verteld wat ons wel en niet gelukt is. Ook hebben we geprobeerd alle problemen eenvoudig uit te leggen aan Royal Smit, zodat ze het beter kunnen begrijpen. Het doel van het verslag is uiteindelijk om een volgend groepje meer informatie en kennis te geven, zodat een nieuw groepje meteen aan de slag kan gaan.