Programming a CNC-machine using ILP

Maarten Bos

Discrete Mathematics and Mathematical Programming
Department of Applied Mathematics
University of Twente

Date:

15-12-2011

Graduation committee:

dr. W. Kern
dr. N. Litvak
prof. dr. M.J. Uetz

Voorwoord

Dit verslag bevat mijn master thesis horende bij mijn master Discrete Wiskunde en Mathematisch Programmeren. Het onderzoek is gedaan in opdracht van het bedrijf Kast op Maat te Hengelo (ov).

Ik wil graag mijn begeleider Walter Kern bedanken voor het een half jaar lang ondersteunen van mijn onderzoek. Daarnaast wil ik de rest van mijn afstudeercommissie bestaande uit Nelly Litvak en Marc Uetz bedanken. Als laatste wil ik Jelle Duives bedanken voor zijn hulp bij het gebruik van CPLEX.

Samenvatting

Het bedrijf Kast op Maat produceert op maat gemaakte kasten. Hierbij wordt gebruik gemaakt van een computer gestuurde freesmachine, die de houten panelen freest waaruit de kasten bestaan. Tijdens het frezen liggen de panelen op een zogenaamd freesbed. Het freesbed bestaat uit acht beweegbare balken met vacuümcups die ervoor zorgen dat het paneel goed op zijn plek blijft tijdens het freesproces. Het probleem is dat de positionering van deze cups voor elk paneel apart bepaald en in de computer ingevoerd moet worden door de persoon die de machine bedient. Aangezien elke kast op maat gemaakt wordt, zijn alle panelen onderling verschillend. Het kost dan ook veel tijd om voor elk paneel een patroon van vacuümcups te creëren en in te voeren.

In dit onderzoek ontwikkelen we een algoritme dat voor alle door Kast op Maat te bewerken panelen een cuppatroon kan creëren. Dit algoritme moet rekening houden met onder andere de afmetingen van het paneel en de bewerkingen die er aan uitgevoerd gaan worden.

We gaan dit continue probleem modelleren in een discreet model. Hierin wordt het paneel gezien als een eindige verzameling punten. Alleen op deze punten kunnen de vacuümcups het paneel vast houden. Samen vormen deze cups een patroon dat het paneel stabiel moet houden tijdens het frezen. Deze patronen gaan we bepalen met behulp van DP, LP en ILP.

Naast dat het gecreëerde patroon het paneel stabiel moet houden, is het van belang dat het vinden van een oplossing niet teveel tijd in beslag neemt. De bruikbaarheid van het algoritme komt in het geding wanneer het berekenen van een patroon te lang duurt. We bekijken daarom of we het probleem efficiënt kunnen oplossen. Dit doen we door te kijken voor welke panelen de bijbehorende LPs geheeltallige oplossingen geven. Als het probleem efficiënt opgelost kan worden, dan kunnen we met zekerheid zeggen dat er snel genoeg een cuppatroon gevonden kan worden.

Inhoudsopgave

1. Inleiding

Het interieurbouwbedrijf Kast op Maat gevestigd te Hengelo (ov) is gespecialiseerd in het vervaardigen van op maat gemaakte kasten. Daarnaast worden voor bedrijven ook andere meubelen gemaakt zoals balies, displays en stands[1]. Het is voor de klant mogelijk om bij Kast op Maat de meubelen zelf vorm te geven. Dit gebeurt met behulp van een computerprogramma. Een voltooid ontwerp wordt door de klant teruggestuurd naar Kast op Maat. Zij zorgen ervoor dat alle panelen op de juiste maat gefreesd worden zodat deze samen het gevraagde meubelstuk vormen. Deze manier van werken zorgt ervoor dat de klant altijd een kast krijgt die volledig aan zijn wensen voldoet. Het nadeel is dat elke kast uniek is zodat er geen vast protocol is voor het frezen van de panelen.

1.1 CNC-machine

De houten panelen die gebruikt worden om meubelstukken van te maken worden gefreesd door een CNC-machine . Dit staat voor *computer numerical control* en is feitelijk een computer gestuurde machine. Voordat de machine een paneel kan frezen, moet er aan de computer doorgegeven worden wat er aan het paneel gedaan moet worden. Dit gebeurt door middel van een barcode die op het paneel geplakt zit. Deze code wordt met een scanner ingelezen in de bij de machine horende computer. Deze computer heeft verbinding met het internet en haalt met behulp van de code de juiste informatie van een bepaalde website af. Op de website staat de afmeting van het paneel en welke bewerkingen er aan uitgevoerd moeten worden. Bij bewerkingen moet onder andere gedacht worden aan het op de juiste grootte frezen, een figuur uit het paneel of het boren van kleine gaatjes aan de zijkant voor deuvels (ook wel kopse boringen genoemd).

Tijdens het frezen van een paneel is het van belang dat deze goed op zijn plek blijft. Een niet goed vastzittend paneel kan gaan bewegen tijdens het frezen, wat lelijke freesranden tot gevolg kan hebben. Om dit te voorkomen heeft de machine een zogenaamd freesbed. Dit bed bestaat uit een aantal vacuümcups waar het paneel bovenop gelegd wordt. Zodra de machine is geactiveerd, ontstaat er een vacuüm tussen het paneel en de cups. Hierdoor blijft het paneel op zijn plaats tijdens het frezen. Een paneel dat goed vastgehouden wordt door de vacuümcups noemen we stabiel. Als de machine klaar is met het paneel dan haalt de persoon die aan de machine werkt,de machineoperator, het paneel van het freesbed af.

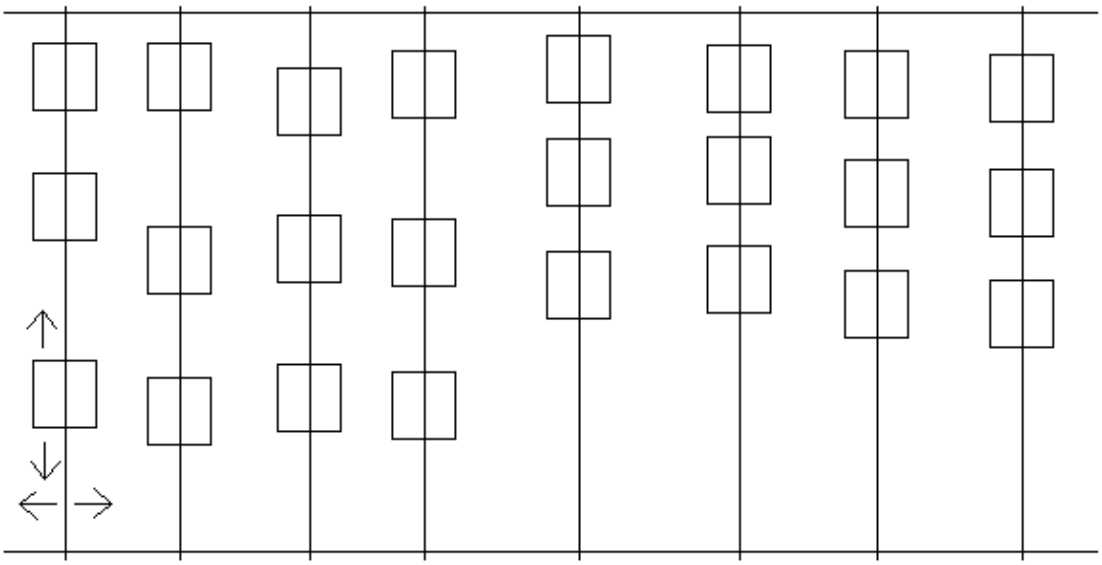
1.2 Vacuümcups

De CNC-machine waar het bedrijf gebruik van maakt heeft een freesbed dat uit 24 vacuümcups bestaat. Deze zitten gelijk verdeeld over acht balken die een gelijke lengte hebben. De balken kunnen parallel aan elkaar bewegen, maar kunnen niet van plaats verwisselen. Dit laatste geldt ook voor de drie cups die zich op elke balk bevinden. In figuur 1.1 is een foto te zien van het een gedeelte van het freesbed. Figuur 1.2 laat een schematisch bovenaanzicht van het freesbed zien.

Figuur 1.1

Vier van de acht balken zijn te zien op deze foto van het freesbed.

Figuur 1.2



Schematisch bovenaanzicht van het freesbed. De acht balken kunnen horizontaal bewegen. De vacuümcups zijn weergegeven als vierkanten en bewegen verticaal.

In figuur 1.1 is te zien dat er verschillende soorten cups gebruikt worden. Zo zijn er cups die even groot zijn als het metalen blok waar ze op liggen. Een voorbeeld hiervan zijn de twee bovenste cups op de meest linker balk. Deze cups worden hele cups genoemd. Er wordt echter ook gewerkt met halve cups. De onderste cup op dezelfde linkerbalk is een halve cup. Het gebruik van halve cups kan handig zijn op posities waar voor een hele cup niet genoeg ruimte is. Al deze cups zitten vastgemaakt aan de metalen blokken, maar kunnen eenvoudig vervangen worden.

De CNC-machine maakt gebruik van een EPS (*electronic positioning system*) voor het positioneren van de vacuümcups. De gewenste locaties van de cups moeten door de machineoperator ingevoerd worden in de computer. Hierbij moet hij rekening houden met de grootte van het paneel en de bewerkingen die de machine gaat uitvoeren. Als een cup geplaatst is op een plek waar ook een gat in het paneel gefreesd moet worden, bestaat de kans dat de machine de cup raakt tijdens het frezen. Hierdoor zal mogelijk het vacuümmechanisme van de cup niet meer werken met als gevolg dat de cup vervangen moeten worden.

De machineoperator moet ervoor zorgen dat genoeg cups het paneel vasthouden tijdens het frezen zodat het paneel op zijn plek blijft liggen. Daarnaast moet hij er rekening mee houden waar de cups zich bevinden die niet actief zijn. Dit kan van belang zijn bij eventuele boringen aan de zijkant van het paneel. Een voorbeeld van deze kopse boringen is het maken van gaten in de zijkant van het paneel om deuvels in te plaatsen. Bij kopse boringen kan de machine vacuümcups die vlak naast het paneel opgesteld staan beschadigen.

1.3 Probleemstelling

De machineoperator moet rekening houden met veel verschillende gegevens, waardoor het bepalen en invoeren van de posities van de vacuümcups lang kan duren. Aangezien de kasten voor de klanten op maat gemaakt worden, en dus alle panelen van elkaar verschillen, moet deze procedure voor elk paneel opnieuw uitgevoerd worden.

Het doel van dit onderzoek is om een algoritme te ontwikkelen dat, op basis van de informatie die de computer krijgt via de barcode, een patroon van vacuümcups te creëren. Zo kan de computer wanneer de informatie ingelezen is direct beginnen met het rekenen aan een cuppatroon. De machineoperator hoeft er dan slechts voor te zorgen dat het paneel op het freesbed gelegd wordt. Dit zal tijd schelen en daarnaast ook de kans verkleinen dat een cup geraakt wordt tijdens het frezen.

Het te ontwerpen algoritme moet op basis van

- De afmetingen van het paneel
- De plekken waar bewerkingen uitgevoerd worden
- De grootte van de vacuümcups

een patroon van cups kunnen creëren zodat het paneel goed vastgehouden wordt tijdens het frezen. Daarnaast moet het algoritme er voor zorgen dat er, op plekken waar kopse boringen uitgevoerd worden, geen cups direct naast het paneel gepositioneerd staan. Verder is het van belang dat het uitvoeren van het algoritme weinig tijd in beslag neemt. Als het algoritme een te grote looptijd heeft, komt de bruikbaarheid ervan in het geding.

1.4 Onderzoeksvragen

Voordat we beginnen met het ontwikkelen van een algoritme, gaan we kijken waar het algoritme precies aan moet voldoen en rekening mee moet houden. Dit doen we aan de hand van vier onderzoeksvragen, die we hier kort toelichten. In paragraaf **1.5** worden deze vragen beantwoord.

(1) Op welke plekken van het paneel kunnen de vacuümcups geplaatst worden?

De vacuümcups moeten ervoor zorgen dat het paneel stabiel blijft tijdens het frezen. Er zijn oneindig veel patronen van deze cups te maken die het paneel vasthouden. Immers zijn twee patronen al verschillend van elkaar wanneer één cup een millimeter verschoven is. Het zoeken naar een patroon is een continu probleem. We gaan dit continue probleem modeleren als een discreet probleem. In deze discretisatie kunnen we duidelijk aangeven waar de cups het paneel kunnen vasthouden.

(2) Wanneer is een paneel stabiel?

Een paneel is stabiel als het door genoeg vacuümcups ondersteund wordt. Dit is echter een te beperkte definitie van stabiliteit. Zo kan een paneel nooit stabiel zijn als alle cups zich aan één uiteinde van het paneel bevinden, ook al zijn dit er dan een groot aantal. Oftewel, de plaats waar de cups het paneel vasthouden is voor de stabiliteit van groot belang. Om hier duidelijkheid in te scheppen gaan we een stabiliteitsvoorwaarde formuleren.

(3) Hoe wordt voorkomen dat cups geraakt worden tijdens het freesproces?

Het is van belang dat bij de plaatsing van de cups, het al bekend is welke bewerkingen er aan een paneel uitgevoerd gaan worden. Met deze bewerkingen kan het algoritme dan rekening houden. We gaan op zoek naar een manier om deze bewerkingen te implementeren in het model. Hiermee kunnen we voorkomen dat er cups kapot gaan doordat ze geraakt worden door de CNC-machine.

(4) Hoelang mag het duren voordat het algoritme een oplossing geeft?

Het voordeel van het algoritme dat we willen ontwikkelen is dat de machineoperator niet zelf de cuppatronen hoeft in te voeren in de computer. Wanneer het algoritme goed werkt betekent dit dat de machine nooit meer een vacuümcup kapot maakt omdat deze in de weg stond. Daarnaast moet het algoritme sneller een patroon kunnen creëren dan dat de machineoperator het in de computer kan invoeren. Dit hangt er wel vanaf hoe lastig het probleem is dat het algoritme op moet lossen. Daarom is het van belang om te kijken naar de complexiteit van het probleem.

1.5 Beantwoorden van de onderzoeksvragen

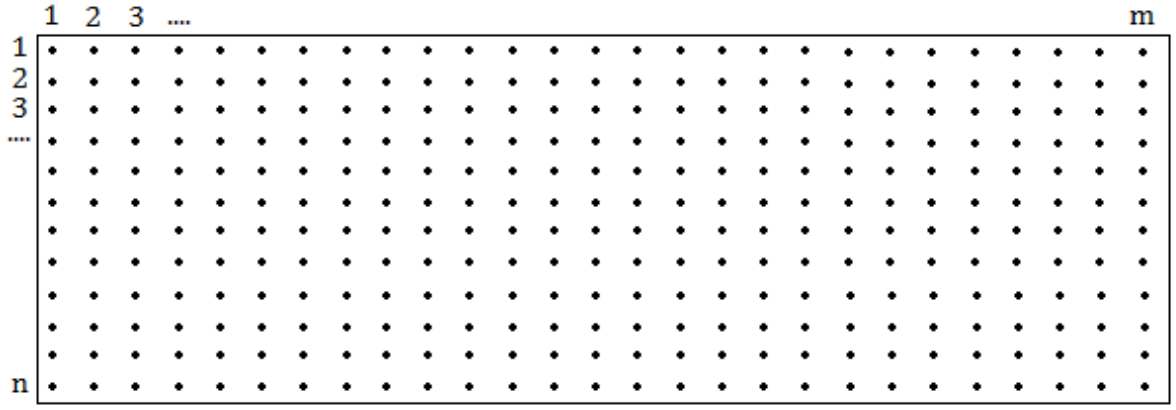
In deze paragraaf gaan we de vier onderzoeksvragen uit **1.4** beantwoorden. Deze antwoorden geven een basis voor het te ontwikkelen algoritme.

(1) *Op welke plekken van het paneel kunnen de vacuümcups geplaatst worden?*

Voordat we een algoritme kunnen ontwerpen dat een patroon van cups maakt, moeten we een model van het probleem maken. Hierin wordt het werkelijke probleem vertaald naar een gemodelleerd probleem zodat er aan gerekend kan worden. Bij het maken van het model hebben we ervoor gekozen om het probleem te discretiseren. Dit houdt in dat bij de oplossing van het probleem (de plaatsen waar een vacuümcup komt te staan) slechts gebruik gemaakt kan worden van een eindig aantal plekken op het paneel. Door genoeg van dit soort plekken toe te staan, zal de oplossing van het gemodelleerde discrete probleem een goede oplossing zijn voor het werkelijke continue probleem. Deze plekken op het paneel waar mogelijk een cup komt te staan noemen we vanaf nu **punten**.

In figuur 1.3 laten we deze discretisatie zien. Hierin is een schematisch bovenaanzicht gegeven van een rechthoekig paneel dat door de CNC-machine bewerkt moet worden. Het rooster van punten dat de discretisatie oplevert is op te delen in rijen en kolommen. Deze zijn aan de zijkant van de figuur genummerd. De panelen liggen over de lengte op het freesbed. De afstand van de onder- tot de bovenkant van het paneel noemen de breedte ervan.

Figuur 1.3



Bovenaanzicht van een rechthoekig paneel. Het paneel is verdeeld in een rooster van punten. Op deze punten kunnen de vacuümcups het paneel vasthouden.

We zullen het in dit onderzoek vooral hebben over rechthoekige panelen, aangezien ongeveer 80% van de panelen die Kast op Maat moet bewerken deze vorm heeft. Bij de resultaten (paragraaf **5.1**) zal blijken dat we met hetzelfde model ook voor niet rechthoekige panelen een cuppatroon kunnen creëren.

(2) *Wanneer is een paneel stabiel?*

De uitkomst van het door ons te maken algoritme moet een patroon van vacuümcups opleveren. Deze cups moeten ervoor zorgen dat het paneel tijdens het frezen goed vastgehouden wordt. Als er genoeg cups zijn die het paneel op de juiste plekken vasthouden, noemen we het paneel stabiel. De vraag in deze is wat genoeg cups zijn en wanneer ze op de juiste plek zitten. We hebben een duidelijke voorwaarde nodig waar ons algoritme rekening mee moet houden. Deze voorwaarde noemen we de **stabiliteitsvoorwaarde**.

In figuur 1.3 hebben we laten zien hoe een paneel verdeeld kan worden in punten; plekken waar een cup geplaatst kan worden. Als een cup op een punt geplaatst wordt, betekent dit dat dit punt goed vastgehouden wordt. De geplaatste cup zorgt er echter voor dat meerdere punten vastzitten. Immers, als dit niet het geval was geweest had elk punt zijn eigen cup nodig gehad. We moeten een maat stellen voor welke punten er nog meer vastgehouden worden door de geplaatste cup. Dit noemen we ook wel het bedekken of coveren van de andere punten. We zeggen dat een punt gecoverd wordt als hij zich binnen een bepaalde maximale afstand van de dichtstbijzijnde cup bevindt. Deze maximale afstand noemen we $\boldsymbol{max}$. De afstand tussen twee punten die naast elkaar liggen stellen we gelijk aan 1. Dit levert de volgende voorwaarde op:

Stabiliteitsvoorwaarde

> *Een paneel is stabiel dan en slechts dan als de afstand van elk punt tot de dichtstbijzijnde cup kleiner of gelijk is aan de maximale afstand.*

Voldoet een patroon niet aan deze conditie dan is het gecreëerde patroon niet stabiel. De andere kant op geredeneerd geldt dit ook. Kort gezegd, de voorwaarde is zowel noodzakelijk als voldoende voor het stabiel zijn van een paneel.

Deze voorwaarde gaat ervan uit dat elke cup evenveel invloed heeft op de stabiliteit van het paneel, of anders gezegd dat elke cup het paneel met dezelfde kracht kan vasthouden. Dit is echter niet het geval, aangezien een halve cup slechts met een kleiner oppervlak het paneel kan vasthouden. We maken hier dan ook de aanname dat er in het cuppatroon alleen hele cups gebruikt worden.

(3) *Hoe wordt voorkomen dat cups geraakt worden tijdens het freesproces?*

De CNC-machine kan verschillende bewerkingen aan een paneel uitvoeren. Zolang deze bewerkingen aan de oppervlakte van het paneel plaatsvinden geeft dit geen verdere restrictie voor de plaatsing van de vacuümcups. Bij een doorboring van het paneel is het echter wel van belang dat er op de plek van een doorboring geen cup geplaatst staat. Als dit wel het geval is, zal de doorboring het vacuüm verbreken. Dit kan de stabiliteit van het paneel in gevaar brengen. Daarnaast is het mogelijk dat hierdoor de vacuümcup kapot gaat.

Om te voorkomen dat er een cup kapot gaat moeten we bij het maken van een patroon rekening houden met eventuele doorboringen van het paneel. Dit doen we door een verzameling van **verboden punten** bij te houden. Bij elke doorboring kijken we welk gedeelte van het paneel daarbij gebruikt wordt. De bijbehorende punten in de discretisatie worden dan verboden punten. Op deze punten is het dan niet mogelijk om een vacuümcup te plaatsen. Het algoritme zal deze punten echter wel moeten coveren om het paneel stabiel te houden.

Op deze manier zorgen we ervoor dat cups niet kapot gaan vanwege doorboringen van het paneel. Daarnaast moeten we ook rekening houden met kopse boringen. Bij het uitvoeren van deze boringen moet er genoeg ruimte aan de zijkant van het paneel zijn zodat de machine daar kan boren. Een kopse boring kan niet uitgevoerd worden wanneer er vlak naast de boring een cup opgesteld staat. We zullen daarom bij het maken van een cuppatroon ook rekening moeten houden met de cups die niet het paneel vasthouden. Dit komt pas ter sprake als we de niet gebruikte cups ook gaan modeleren, wat we in hoofdstuk **4** gaan doen.

(4) *Hoelang mag het duren voordat het algoritme een oplossing geeft?*

Het is belangrijk dat we een algoritme ontwikkelen dat niet teveel tijd nodig heeft om tot een oplossing te komen. Het zou geen verbetering van de situatie zijn als de computer voor elk paneel een paar minuten nodig heeft om een patroon te berekenen. We proberen dit te voorkomen door naar de complexiteitsgraad van het algoritme te kijken. Dit is de manier waarop het algoritme zich gedraagt als het op te lossen probleem toeneemt.

Simpel gezegd kunnen we twee soorten problemen onderscheiden; makkelijke en moeilijke problemen. Makkelijke problemen kunnen we met een algoritme efficiënt, in polynomiale tijd, oplossen. Dit betekent dat de tijd die het algoritme erover doet om tot een oplossing te komen, begrensd is door een polynoom dat afhankelijk is van de grootte van de invoer.

Voor moeilijke problemen is een efficiënt algoritme nog niet gevonden. De tijd die het algoritme nodig heeft om tot een oplossing te komen is in dat geval niet begrensd door een polynoom. Dit betekent dat als de grootte van het probleem toeneemt, dit kan leiden tot een exponentieel langere looptijd van het algoritme.

In dit onderzoek gaan we een algoritme ontwikkelen dat het cuppatroon probleem oplost. Hierbij kijken we of dit algoritme efficiënt het probleem kan oplossen. Als dit het geval is zal het voor de computer geen probleem zijn om snel genoeg tot een oplossing te komen. Wanneer we het probleem met een algoritme niet efficiënt kunnen oplossen moeten we testen hoe snel het algoritme tot een oplossing komt. Als blijkt dat onder bepaalde omstandigheden de looptijd van het algoritme te groot is, gaan we kijken naar benaderingsalgoritmes.

In dit onderzoek gaan we ervan uit dat het cuppatroon ter plekke bepaald moet worden door de computer van de CNC-machine. Dit betekent wel dat deze computer uitgerust moet zijn met de juiste software om voor elk paneel het cuppatroon probleem op te kunnen lossen. Er is nog een andere optie die misschien meer voor de hand ligt.

De computer haalt via een barcode de informatie over het te bewerken paneel van een bepaalde website. Op deze manier weet de computer wat de afmetingen van het paneel zijn en wat er aan gefreesd moet worden. Echter zou via deze website ook meteen doorgegeven kunnen worden in welk patroon de cups moeten staan om het desbetreffende paneel stabiel te houden. De software hoeft in dat geval alleen op een centrale server aanwezig te zijn. Het voordeel hiervan is dat niet elke computer van een CNC-machine apart die software hoeft te hebben. Daarnaast scheelt dit ook tijd bij het bewerken van de panelen, aangezien het patroon niet meer ter plekke berekend hoeft te worden. In dit geval zou de tijd die het algoritme nodig heeft om tot een oplossing te komen ook minder van belang zijn, aangezien de server ruim van tevoren weet welke panelen er moeten worden bewerkt.

Dit zou betekenen dat het algoritme er langer over mag doen om tot een oplossing te komen. Wij gaan er echter van uit dat het patroon niet van een server afgehaald kan worden. Het uitvoeren van het algoritme mag daarom niet teveel tijd in beslag nemen. In dit geval stellen we de toelaatbare looptijd van het algoritme op 5 seconden.

Conclusie

Door het te bewerken paneel te discretiseren hebben we een continu probleem vertaald naar een discreet model. De punten in de discretisatie geven aan waar de vacuümcups het paneel kunnen vasthouden. Door middel van een stabiliteitsvoorwaarde kunnen we vaststellen wanneer een paneel goed vastgehouden wordt door de vacuümcups. Daarnaast voorkomen we dat de freesmachine bij een doorboring een cup raakt, door een verzameling van verboden punten bij te houden. Ten slotte gaan we er bij het ontwikkelen van het algoritme op letten of het probleem efficiënt opgelost kan worden. Wanneer dit niet het geval is moeten we misschien een benaderingsalgoritme gebruiken om de oplossing van het probleem te benaderen. Op de hierboven beschreven antwoorden van de onderzoeksvragen kunnen we ons algoritme baseren. In hoofdstuk **2** beginnen we met het creëren van het algoritme.

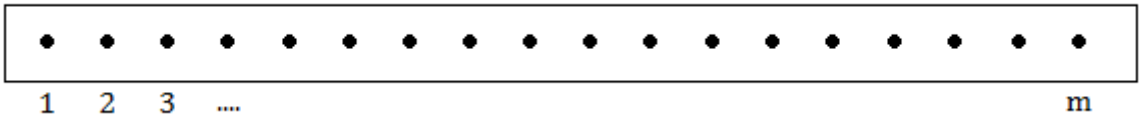
2. Een simpel voorbeeld

In dit hoofdstuk maken we een begin met het algoritme door te kijken naar een simpel voorbeeld. We laten zien hoe de discretisatie van dit voorbeeld eruit ziet en we kijken hoe we voor dit paneel een cuppatroon kunnen berekenen. Hiervoor gebruiken we dynamisch en lineair programmeren. Daarnaast gaan we kijken naar de complexiteit van het probleem.

2.1 Probleemomschrijving

Het paneel dat we in dit hoofdstuk gaan bekijken is een rechthoekig paneel. Om het simpel te houden kijken we voor nu alleen naar panelen die een kleine breedte hebben. Deze panelen hebben een dusdanig kleine breedte dat slechts één vacuümcup per balk het paneel kan vasthouden. Het is dus niet mogelijk om twee cups boven elkaar het paneel vast te laten houden. Dit betekent dat in de discretisatie van het paneel alle punten zich in één rij bevinden. Figuur 2.1 laat deze discretisatie zien.

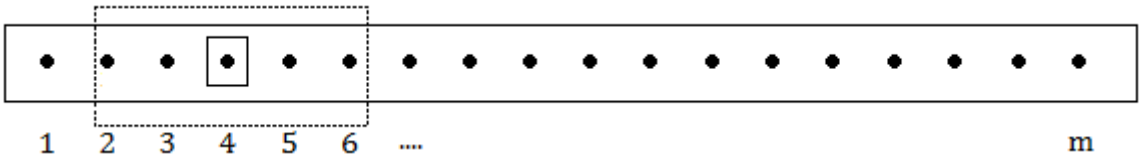
Figuur 2.1



Schematisch bovenaanzicht van een rechthoekig paneel met een kleine breedte. De stippen zijn de punten waar een cup geplaatst kan worden. Door de geringe breedte is het niet mogelijk twee cups boven elkaar te plaatsen.

Deze punten stellen de plekken voor waar een cup het paneel kan vasthouden. Van alle punten moeten we een deelverzameling selecteren waar cups op geplaatst gaan worden. Samen vormen deze cups een patroon dat het paneel stabiel moet houden. Dit patroon moet voldoen aan de stabiliteitsvoorwaarde zoals uitgelegd in paragraaf **1.5**. Oftewel, de geplaatste cups moeten alle punten van de discretisatie coveren. Figuur 2.2 laat grafisch zien wat de stabiliteitsvoorwaarde voor dit smalle paneel betekent.

Figuur 2.2



Het smalle paneel bevat m punten. In dit voorbeeld is max gelijk aan 2. Het vierkantje om punt 4 is een geplaatste cup. De stippellijn geeft aan welke punten door deze cup gecoverd worden. Punt 1 ligt meer dan de maximale afstand van de cup af en wordt op dit moment niet gecoverd.

Het algoritme moet een aantal punten selecteren zodat het paneel stabiel gehouden wordt wanneer er op deze punten cups geplaatst worden. Daarnaast kiezen we ervoor om het algoritme een extra eis mee te geven. Het algoritme moet het patroon vinden dat het minste aantal balken gebruikt. Dit heeft twee duidelijke voordelen. Allereerst geeft het minimaliseren van het aantal balken een scheiding van panelen die wel en niet stabiel gehouden kunnen worden op het freesbed. Als in de oplossing van het algoritme meer dan acht balken gebruikt worden, dan kan het paneel op dit freesbed niet stabiel gehouden worden. Naar onze eisen is het dan niet mogelijk om het paneel op het freesbed in kwestie te bewerken.

Verder is het mogelijk om meerdere panelen naast elkaar op het freesbed te plaatsen als hier ruimte voor is. Het voordeel hiervan is dat de CNC-machine meteen door kan met het frezen van het tweede paneel wanneer het klaar is met het eerste. Door het aantal balken te minimaliseren kan hier optimaal gebruik van worden gemaakt. Vanaf nu noemen we een patroon van cups **optimaal** als deze gebruik maakt van een minimum aantal balken.

In paragraaf **2.2** zal het probleem zodanig geformuleerd worden dat we met behulp van dynamisch programmeren een optimaal cuppatroon kunnen verkrijgen.

2.2 Dynamisch Programmeren

Dynamisch programmeren is een oplosmethode waarbij problemen opsplitst worden in makkelijkere subproblemen. Bij ons probleem betekent dit dat we niet in één keer een optimaal patroon proberen te vinden voor het hele paneel, maar dat we per punt kijken wat het beste is om te doen: wel of geen cup plaatsen. Op deze manier kunnen we van het ene eind van het paneel (punt m) naar het andere eind (punt 1) gaan en een beslissing maken bij elk punt dat we tegenkomen. Dit levert een recursieve formule op, waarin we de waarde van het probleem (het aantal balken dat we gebruiken in een patroon) minimaliseren. De gemaakte beslissingen geven de posities van de cups in het patroon weer.

Het dynamisch programma horende bij ons probleem heeft de volgende eigenschappen:

- Punten $j = \{1,2, \dots, m\}$ op het paneel waar een cup neergezet kan worden. Bij het dynamisch programmeren moet er op deze punten een beslissing genomen worden.
- A_j is de beslissingsvariabele van punt j . $A_j = 1$ als er een cup geplaatst wordt, $A_j = 0$ als dit niet het geval is. Wanneer j een verboden punt is, dan volgt automatisch $A_j = 0$
- max is de maximale toegestane afstand van een punt tot de dichtstbijzijnde cup. Dit zorgt ervoor dat het paneel stabiel gehouden wordt.
- k is de afstand van punt j tot de volgende geplaatste cup links van het punt j (op punt $j - k$). Als $k = 2 * max + 1$ (dit is de grootste afstand tussen twee geplaatste cups waarbij alle tussenliggende punten gecoverd worden) dan moet er een cup geplaatst worden, oftewel $A_j = 1$. k' is de afstand tot de volgende cup in het punt $j + 1$.
- Waarde V_j van punt j. Dit is het aantal balken dat al gebruikt is in het patroon op de punten $\{j, j + 1, j + 2, \dots, m\}$.

We gaan nu gebruik maken van dynamisch programmeren. Het minimaliseren over de mogelijke beslissingen in de volgende recursieve formule levert een optimale plaatsing van de cups op.

$$V_j(k) = \min\{A_j + V_{j+1}(k')\}$$

Zoals gezegd kan beslissingsvariabele A_j slechts twee waarden aannemen (1 of 0 voor het plaatsen van wel of geen cup). Hierdoor kan de recursieve formule als volgt uitgeschreven worden:

$$V_j(k) = \min\{1 + V_{j+1}(1), 0 + V_{j+1}(k+1)\}, \qquad k \in \{1,2,\dots,2 * max\}$$

$$V_j(k) = 1 + V_{j+1}(1), \qquad k = 2 * max + 1$$

Deze recursieve formule gebruiken we bij alle punten op het paneel behalve de twee randpunten ($j = 1$ en $j = m$). Daarvoor stellen we aparte vergelijkingen op. Als eerste kijken we naar de startwaarde ($j = m$). Als we bovenstaande formule zouden volgen, moeten we kijken naar de waarde in het punt $j + 1$ en dan besluiten om wel of geen cup te plaatsen. Dit punt bestaat echter niet omdat we in dit geval aan het uiteinde van het paneel zitten. Voor de startwaarde kijken we dus alleen naar het punt m en kiezen dan de beste beslissing. We onderscheiden hier twee gevallen. Wanneer er binnen de maximale afstand links van het punt m een cup wordt geplaatst, hoeft er op punt m zelf geen cup geplaatst te worden. Dit moet alleen als de dichtstbijzijnde cup verder dan max links van punt m ligt.

Dit levert de volgende startwaarden op:

$$V_m(k) = 0, \qquad k \in \{1,2,\dots,max\}$$

$$V_m(k) = 1, \qquad k = max + 1$$

Ook voor het punt ($j = 1$) moeten we de recursieve formule aanpassen. In dit geval is het bestaan van punt $j + 1$ niet het probleem. De waarde van dit punt is echter afhankelijk van de afstand tot de dichtstbijzijnde cup links van dit punt. Dit kan problemen opleveren, aangezien er geen punten meer links van het punt 1 zijn. Dit lossen we op door een virtueel punt links van het paneel aan te maken. Dit punt zetten we op een afstand van $(max + 1)$ links van punt 1 neer en zetten daar ook een cup op. Het gevolg hiervan is dat de afstand tot de dichtstbijzijnde cup rechts van punt 1 altijd kleiner dan of gelijk is aan max. Dit aangezien de maximale afstand tussen twee cups gelijk is aan $2 * max + 1$. Hiermee zorgen we ervoor dat ook $j = 1$ gecoverd wordt door een cup.

Dit levert de volgende formule op voor het punt $j = 1$.

$$V_1(max + 1) = \min\{1 + V_2(1), 0 + V_2(max + 2)\}$$

De oplossing hiervan geeft ons het aantal balken dat we nodig hebben (de waarde V_1) alsmede de punten waar de cups geplaatst worden; alle punten j waarvoor geldt dat $A_j = 1$.

2.3 Het LP model

Het *set cover* probleem is een bekend probleem in de wiskunde, wat als volgt is gedefinieerd:

Gegeven is een universum U met elementen $u = \{1,2,3, \dots, m\}$ en een aantal verzamelingen $S \in \mathcal{S},\ S \subseteq U$ wiens vereniging het gehele universum bevat. Bij het set cover probleem wordt er gekeken naar een minimum aantal verzamelingen wiens vereniging alle elementen van het universum bevat. Dit kan geschreven worden in de vorm van een *integer linear program* (ILP). Hierin is x_S de variabele die de verzamelingen toewijst aan de cover; $x_S = 1$ wanneer verzameling S deel uit maakt van de cover.

$$\min \sum_{S \in \mathcal{S}} x_S \qquad \text{(minimaliseer het aantal verzamelingen)}$$

$$s.t. \sum_{S:\, u \in S} x_S \geq 1, \qquad \forall\, u \in U \qquad \text{(elk element in het universum wordt gecoverd)}$$

$$x_S \in \{0,1\}, \qquad \forall\, S \in \mathcal{S} \qquad \text{(elke verzameling zit wel of niet in de cover)}$$

Ons cupprobleem kan op eenzelfde manier gemodelleerd worden, waarbij we het vertalen naar het set cover probleem. In dat geval zijn de punten op het paneel de elementen waar het universum uit bestaat. De verzamelingen $S \in \mathcal{S}$ zijn de punten die gecoverd worden door het plaatsen van een cup op een bepaald punt. Zo bevat verzameling S_1 alle punten die gecoverd worden als er een cup geplaatst wordt op punt 1. Stel dat $max = 2$, dan geldt $S_1 = \{1,2,3\}$, aangezien deze punten zich binnen de maximale afstand van de cup op het punt 1 bevinden.

Hieruit blijkt dat ons probleem een speciaal geval is van het set cover probleem. We kunnen ons probleem formuleren als een ILP dat de volgende eigenschappen heeft:

- Punten $j = \{1,2,3, \dots, m\}$ waar een cup geplaatst kan worden.
- $x_j = 1$ als een cup geplaatst wordt op het punt j. $x_j = 0$ als dit niet het geval is.
- $a_{ij} = 1$ als punt j zich binnen de maximale afstand van punt i bevindt. Met andere woorden $a_{ij} = 1$ als punt i gecoverd kan worden door punt j. Als punt l verboden is geldt $a_{il} = 0, \ \forall i$.

$$\min \sum_j x_j \qquad \text{(minimaliseer het aantal balken)}$$

$$s.t. \sum_j a_{ij}\, x_j \geq 1, \qquad \forall i \qquad \text{(alle punten worden gecoverd; stabiliteitsvoorwaarde)}$$

$$x_j \in \{0,1\}, \qquad \forall j \qquad \text{(elk punt krijgt wel of geen cup)}$$

Het is bekend dat de beslissingsvariant van het algemene set cover probleem NP-volledig is en de optimalisatievariant NP-moeilijk. Voor deze problemen zijn er geen efficiënte algoritmes bekend. Ons probleem is echter niet gelijk aan het set cover probleem maar een speciaal geval ervan.

We zien dat de verzamelingen $S \in \mathcal{S}$ horende bij dit probleem een bepaalde structuur hebben. Zo zijn alle verzamelingen even groot (behalve aan de randen van het paneel). Ook zijn de punten in een verzameling opeenvolgend. Hiermee bedoelen we dat als punt $j - 1$ en $j + 1$ in een bepaalde verzameling zitten, dan moet punt j daar ook deel van uitmaken. We kunnen bewijzen dat deze eigenschappen ervoor zorgen dat het probleem efficiënt op te lossen is. We gebruiken bij dit bewijs de volgende definities.

Definitie 2.1

*Een vierkante matrix $A = [a_{ij}]$ is **unimodulair** als alle a_{ij} geheeltallig zijn en als de determinant van A gelijk is aan +1 of -1.*

Definitie 2.2

*Een matrix A is **totaal unimodulair** als elke vierkante niet-singuliere deelmatrix van A unimodulair is.*

Deze definities worden gebruikt in stelling **2.2**. Bij het bewijs van deze stelling gaan we de regel van Cramer toepassen. In stelling **2.1** wordt deze zonder bewijs gegeven.

Stelling 2.1 (Regel van Cramer)

Stel dat $\{Ax \leq b\}$ *een stelsel van* n *vergelijkingen met* n *onbekenden is en waarin matrix* A *inverteerbaar is. De oplossing van dit stelsel* $x = (x_1, x_2, \dots, x_n)$ *kan bepaald worden met* $x_i = \frac{\det(A_i)}{\det(A)}$. *Hierin is* A_i *de matrix die ontstaat door kolom* i *van matrix* A *te vervangen door vector* b.

Stelling 2.2

Stel dat matrix A *totaal unimodulair is en dat vector* b *bestaat uit gehele getallen, dan is de oplossing* $\{x|Ax \leq b\}$ *geheeltallig.*

Bewijs:

Stel matrix A heeft een grootte van m x n. De oplossing x' van het stelsel $\{Ax \leq b\}$ geeft n lineair onafhankelijke ongelijkheden die strikt zijn. Dit kunnen we schrijven als het stelsel $\{A'x' = b'\}$ waarin A' uit deze n vergelijkingen bestaat en b' de bij deze vergelijkingen horende invoer uit vector b is. We gebruiken de regel van Cramer om deze oplossing x' te berekenen. Die zegt dat $x'_i = \frac{\det(A'_i)}{\det(A')}$. Van $\det(A'_i)$ weten we dat deze geheeltallig is aangezien de matrix A'_i, $\forall i$ uit alleen maar gehele getallen bestaat. Verder weten we dat matrix A totaal unimodulair is, wat betekent dat elke vierkante submatrix een determinant heeft van 0,-1,+1. In dit geval is $\det(A') \neq 0$ aangezien matrix A' bestaat uit n lineair onafhankelijke vergelijkingen. Hieruit volgt dat $\det(A')$ gelijk is aan -1 of +1. Dit betekent dat x' een geheeltallige oplossing is.

Voordat we stelling **2.2** kunnen toepassen op ons probleem, moeten we aantonen dat de restrictiematrix A uit het ILP model totaal unimodulair is. Hiervoor gebruiken we de volgende twee stellingen, waarbij we stelling **2.3** gebruiken om **2.4** te bewijzen.

Stelling 2.3

De node-edge incidence matrix van een gerichte graaf is totaal unimodulair.

Het bewijs van stelling **2.3** is te vinden in [2].

Stelling 2.4

Stel dat matrix $A = [a_{ij}]$ een binaire matrix is; $a_{ij} \in \{0,1\}$. Als in elke rij van A de enen opeenvolgend voorkomen, dan is matrix A totaal unimodulair.

Bewijs:

Stel dat we een $m \times n$ binaire matrix $A = [a_{ij}]$ hebben. Hiervan nemen we een $t \times t$ submatrix B. Net als in matrix A zijn in matrix B in elke rij de enen opeenvolgend. Daarnaast introduceren we een $t \times t$ matrix N die er als volgt uit ziet:

$$N = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

We kunnen dan matrix N en matrix B^T met elkaar vermenigvuldigen. Het resultaat van deze vermenigvuldiging NB^T heeft in elke kolom naasten nullen maximaal één +1 en één -1. Dit betekent dat matrix NB^T een submatrix is van een *node-edge incidence* matrix. Uit stelling **2.3** weten we dat deze matrices totaal unimodulair zijn, wat betekent dat de determinant ervan gelijk moet zijn aan $\{0, -1, +1\}$. Aangezien de determinant van matrix N gelijk is aan 1, geeft dit dat $\det(B^T) = \det(B) \in \{0, -1, +1\}$. Door gebruik te maken van definities **2.1** en **2.2** kunnen we concluderen dat matrix A totaal unimodulair is.

De restrictiematrix $A = [a_{ij}]$ in ons model is een binaire matrix. Stelling **2.4** is toepasbaar op A vanwege de opeenvolgende verzamelingen van punten die gecoverd worden bij het plaatsen van een cup. Dit is terug te zien in matrix A als opeenvolgende enen in elke kolom en rij. Hieruit kunnen we concluderen dat A totaal unimodulair is. Verder is vector b in ons geval een vector bestaande uit enkel enen. Dit betekent dat stelling **2.2** toepasbaar is op ons stelsel van vergelijkingen.

Uit stelling **2.2** volgt dat de oplossing van het stelsel altijd een geheeltallige is. Dit betekent dat we de geheeltalligheidsrestrictie in het ILP minder streng kunnen maken zonder dat de oplossing van het probleem daardoor verandert. Dit wordt ook wel het *relaxen* van de restrictie genoemd. In plaats van alleen een waarde van nul of één toe te laten, accepteren we het ook als een variabele x_j daartussen ligt. Deze restrictie ziet er dan als volgt uit:

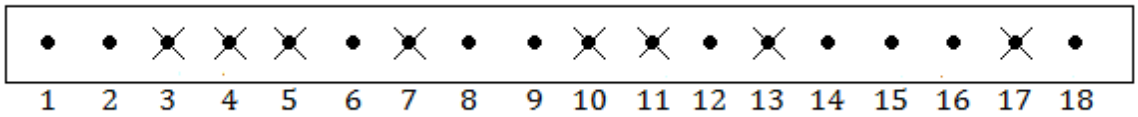
$$0 \leq x_j \leq 1, \qquad \forall j$$

We hebben nu het ILP geschreven als een LP. Het voordeel hiervan is dat een LP efficiënt opgelost kan worden met behulp van een LP solver.

Voorbeeld

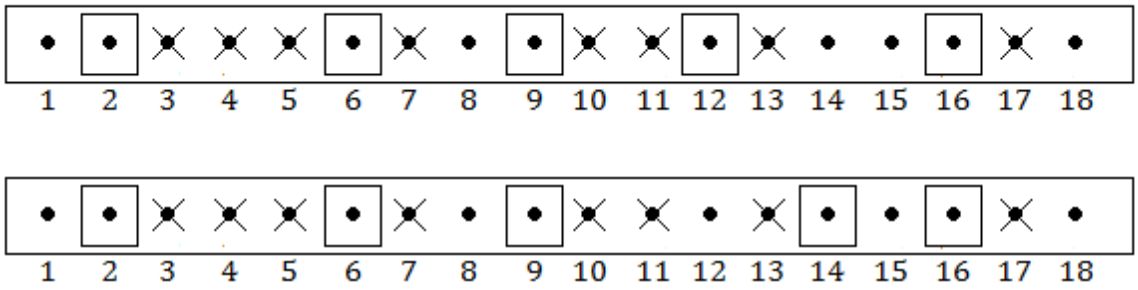
Hier volgt een voorbeeld van de gepresenteerde methodes. In figuur 2.3 laten we de discretisatie van het paneel zien waar we een cuppatroon voor gaan creëren. Deze discretisatie bestaat uit 18 punten, waarvan een aantal verboden is. Deze verboden punten zijn weergegeven als kruisen. We stellen dat de afstand van elk punt tot de dichtstbijzijnde cup maximaal 2 mag zijn. We gaan met zowel dynamisch programmeren als met lineair programmeren een patroon voor dit paneel berekenen. De uitwerkingen hiervan zijn te zien in bijlagen **A** en **B**. Hoe deze oplossingen eruit zien, is weergegeven in figuur 2.4. Hier zijn de geplaatste cups te zien als vierkanten die om de punten heen staan.

Figuur 2.3



Het paneel waarvoor we een patroon gaan bepalen. De verboden posities zijn te zien als kruizen.

Figuur 2.4



Twee mogelijke oplossingen van het cupprobleem. Beide zijn optimaal en hebben 5 balken nodig om het paneel stabiel te houden tijdens het frezen.

Conclusie

In dit hoofdstuk hebben we gekeken naar een simpel voorbeeld van een te bewerken paneel. Door middel van dynamisch programmeren hebben we voor dit paneel een optimaal cuppatroon kunnen creëren. Verder hebben we laten zien dat dit probleem te schrijven is als een speciaal geval van het set cover probleem. Met behulp van totale unimodulariteit hebben we aangetoond dat het cuppatroon probleem met een LP solver efficiënt opgelost kan worden. In het volgende hoofdstuk breiden we ons model uit met panelen die met twee cups boven elkaar vastgehouden kunnen worden.

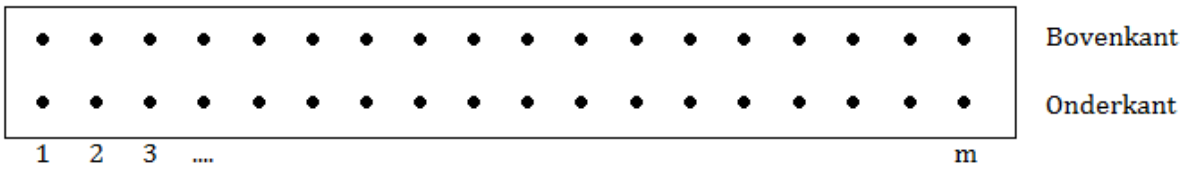
3. Twee cups per balk

In het vorige hoofdstuk hebben we gekeken naar optimale cuppatronen voor smalle panelen. Deze panelen vormen maar een klein gedeelte van alle panelen die bewerkt moeten worden. In dit hoofdstuk gaan we het model uitbreiden zodat ook voor bredere panelen een patroon van vacuümcups gecreëerd kan worden.

3.1 Discretisatie

De panelen waar we in dit hoofdstuk naar gaan kijken zijn breder en kunnen per balk door twee cups vastgehouden worden. In de discretisatie van deze panelen is dit terug te zien als twee rijen punten die boven elkaar staan. Om aan te geven welke punten we precies bedoelen, zullen we naast de nummering de termen boven- en onderkant van het paneel gebruiken. In figuur 3.1 is de discretisatie van dit paneel te zien.

Figuur 3.1

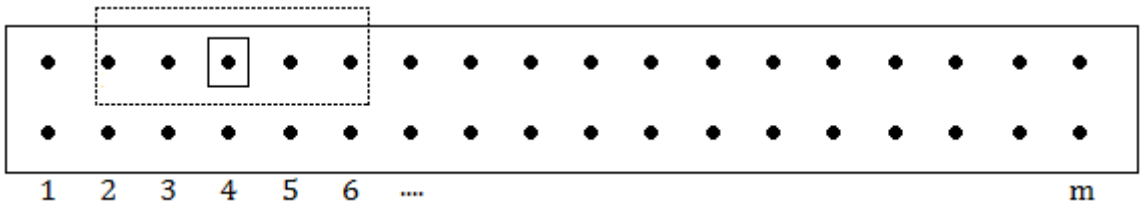


Discretisatie van een paneel waar twee cups boven elkaar mogelijk zijn.

Voordat we voor dit paneel een cuppatroon gaan creëren, kijken we naar de stabiliteitsvoorwaarde. In deze voorwaarde, zoals die geformuleerd is in paragraaf **1.5**, staat dat het paneel stabiel is als elk punt zich binnen de maximale afstand van een cup bevindt. Bij het paneel dat te zien is in figuur 3.1 zou dit betekenen dat een cup die bijvoorbeeld aan de bovenkant van het paneel is geplaatst ook punten aan de onderkant van het paneel zou coveren.

In dit hoofdstuk maken we echter een kleine aanpassing aan de stabiliteitsvoorwaarde. Wij gaan er hier vanuit dat een cup geplaatst aan één kant van het paneel geen cups kan coveren aan de andere kant van het paneel. Deze simplificatie van de stabiliteitsvoorwaarde is te zien in figuur 3.2. Later in hoofdstuk **4** zullen we de panelen die we in dit hoofdstuk bespreken ook bekijken wanneer ze aan de originele stabiliteitsvoorwaarde moeten voldoen.

Figuur 3.2



De geplaatste cup laat zien welke punten er gecoverd worden bij een maximale afstand van twee. De cup in de bovenste rij heeft geen invloed op de punten in de onderste rij.

3.2 Dynamisch Programmeren

In het vorige hoofdstuk hebben we met behulp van dynamisch programmeren een optimaal patroon van cups gecreëerd. We gaan deze methode hier opnieuw toepassen. Doordat er nu zowel aan de boven- als onderkant van het paneel cups geplaatst kunnen worden, zijn ten opzichte van het vorige probleem de recursieve formules aangepast. Bij het maken van een cuppatroon beginnen we nog steeds aan één uiteinde van het paneel en maken dan bij elke punt wat we tegenkomen een beslissing. Bij deze panelen komen we dan echter niet één maar twee punten tegen die boven elkaar liggen. Per beslissing nemen we een besluit voor beide punten.

Het probleem heeft de volgende eigenschappen:

- Punten $j = \{1,2,3, \dots, m\}$ waar een beslissing voor twee punten genomen moet worden.
- A_j is de beslissingsvariabele op het punt j. $A_j = 1$ als er een balk geplaatst wordt, $A_j = 0$ als dit niet het geval is.
- Waarde V_j in het punt j is gelijk aan het aantal balken dat geplaatst is op de punten $\{j, j+1, j+2, \dots, m\}$.
- k is de afstand van punt j tot de volgende geplaatste cup (op punt $j-k$) aan de bovenkant van het paneel. Als $k = 2 * max + 1$ moet er een cup geplaatst worden, oftewel $A_j = 1$.
- l is de afstand van punt j tot de volgende geplaatste cup (op punt $j-l$) aan de onderkant van het paneel. Als $l = 2 * max + 1$ moet er een cup geplaatst worden, oftewel $A_j = 1$.
- k' en l' zijn de afstanden in het punt $j+1$.

We gebruiken bijna dezelfde recursieve formule als die we hebben gebruikt bij het smalle paneel. Hier is de waardefunctie V_j echter afhankelijk van zowel de afstand aan de boven- als aan de onderkant van het paneel tot de eerst volgende cup.

$$V_j(k,l) = \min\{A_j + V_{j+1}(k', l')\}$$

Deze formule gaan we minimaliseren over alle mogelijke beslissingen A_j. Waar in het vorige probleem slechts twee beslissingen mogelijk waren, zijn dat er nu vier. Deze beslissingen zijn het plaatsen van een cup aan: de bovenkant, de onderkant, beide kanten of geen van beide kanten. In het laatste geval hoeft er ook geen balk geplaatst te worden $(A_j = 0)$. Bij de eerste drie is dit wel noodzakelijk $(A_j = 1)$, wat leidt tot het toenemen van de waardefunctie.

Het uitschrijven van de beslissingen resulteert in de volgende recursieve formules:

$$V_j(k,l) = \min\begin{Bmatrix} 1 + V_{j+1}(1,1), 1 + V_{j+1}(1, l+1), \\ 1 + V_{j+1}(k+1,1)\ , 0 + V_{j+1}(k+1, l+1) \end{Bmatrix}, \qquad k, l \in \{1,2, \dots, 2 * max\}$$

$$V_j(k,l) = \min\{1 + V_{j+1}(1,1), 1 + V_{j+1}(1, l+1)\}, \qquad l < k = 2 * max + 1$$

$$V_j(k,l) = \min\{1 + V_{j+1}(1,1), 1 + V_{j+1}(k+1,1)\}, \qquad k < l = 2 * max + 1$$

$$V_j(k,l) = 1 + V_{j+1}(1,1), \qquad k = l = 2 * max + 1$$

Net als in het vorige hoofdstuk moeten we deze recursieve formules aanpassen voor de randen van het paneel. Dit doen we voor zowel de startwaarden ($j = m$) als de voor de eindwaarden ($j = 1$) op dezelfde manier zoals in paragraaf **2.2** beschreven is.

Startwaarden ($j = m$) :

$$V_m(k,l) = 0, \qquad k,l \in \{1,2,\dots,max\}$$

$$V_m(k,l) = 1, \qquad k = max+1 \ \text{ of } \ l = max+1$$

Formule voor het eindpunt ($j = 1$) :

$$V_1(max+1, max+1) = \min\begin{Bmatrix} 1 + V_2(1,1), 1 + V_2(1, max+2), \\ 1 + V_2(max+2,1)\ ,0 + V_2(max+2, max+2) \end{Bmatrix}$$

De oplossing van deze laatste vergelijking geeft ons het aantal balken dat gebruikt wordt in het cuppatroon ($V_1(max+1, max+1)$). De waarden van de beslissingsvariabelen A_j geven de posities van de geplaatste balken en cups.

3.3 Het LP model

We hebben in het vorige hoofdstuk kunnen zien dat het mogelijk is om het probleem te formuleren als een ILP. We konden zelfs aantonen dat het probleem altijd een geheeltallige oplossing had, zodat de geheeltalligheidsrestrictie overbodig was. Of dit ook het geval is met het huidige probleem gaan we in deze paragraaf bekijken. We schrijven het probleem op als een ILP met de volgende eigenschappen:

- Punten $j = \{1,2,3,\dots,m\}$ waar een balk geplaatst kan worden.
- $x_{1j} = 1$ als een cup alleen aan de bovenkant van punt j geplaatst wordt.
- $x_{2j} = 1$ als een cup alleen aan de onderkant van punt j geplaatst wordt.
- $y_j = 1$ als cups aan zowel de boven- als de onderkant geplaatst worden in punt j.
- $a_{ij} = 1$ als punt j punt i covert aan de bovenkant van het paneel.
- $b_{ij} = 1$ als punt j punt i covert aan de onderkant van het paneel.
- Als l een verboden punt is aan de bovenkant van het paneel, dan volgt $a_{il} = 0, \forall i$.
- Als l een verboden punt is aan de onderkant van het paneel, dan volgt $b_{il} = 0, \forall i$.

$$\min \sum_j x_{1j} + x_{2j} + y_j \qquad \text{(minimaliseer het aantal balken)}$$

$$s.t. \sum_j a_{ij}\left(x_{1j} + y_j\right) \geq 1, \qquad \forall i \qquad \text{(alle punten aan de bovenkant zijn gecoverd)}$$

$$\sum_j b_{ij}\left(x_{2j} + y_j\right) \geq 1, \qquad \forall i \qquad \text{(alle punten aan de onderkant zijn gecoverd)}$$

$$x_{1j},\ x_{2j},\ y_j \in \{0,1\}, \qquad \forall j \qquad \text{(elk punt krijgt wel of geen cup)}$$

We zijn vooral geïnteresseerd in de laatste geheeltalligheidsrestrictie, waarbij we gaan kijken of we deze restrictie kunnen relaxen zonder dat we een niet-geheeltallige oplossing krijgen. Als dit het geval is, kan met behulp van een LP solver dit probleem efficiënt opgelost worden. Alvorens we dit gaan bekijken, herschrijven we het probleem zodat alle restricties van zowel de boven- als onderkant van het paneel in één matrix staan. Deze matrix noemen we matrix C en ziet er als volgt uit.

$$C = \begin{bmatrix} A & 0 & A \cap B \\ 0 & B & A \cap B \end{bmatrix},$$

$$x = [x_1\ x_2\ y]^{\mathsf{T}},$$

waarbij $A = [a_{ij}]$ en $B = [b_{ij}]$. Omdat de maximale afstand zowel aan de boven- als de onderkant gelijk is aan max, zijn de matrices A en B gelijk aan elkaar zolang we de verboden punten niet in acht nemen. De matrix $A \cap B$ is de restrictiematrix horende bij de variabele y. Die staat voor het plaatsen van cups aan beide kanten van het paneel. Dit zou niet mogelijk moeten zijn wanneer één van de twee kanten op die plek een verboden punt heeft. Vandaar dat als één van de twee matrices A en B een verboden punt heeft, $A \cap B$ bij dat punt een nulkolom heeft. Wanneer er aan beide kanten geen verboden punten zijn, dan geldt $A \cap B = A = B$. Dit levert de volgende ILP formulering op:

$$\min \sum_j x_j \qquad \text{(minimaliseer het aantal balken)}$$

$$s.t.\ Cx \geq 1 \qquad \text{(alle punten worden gecoverd, stabiliteitsvoorwaarde)}$$

$$x_j \in \{0,1\}, \qquad \forall j \qquad \text{(elk punt krijgt wel of geen cup)}$$

3.4 Geheeltalligheid

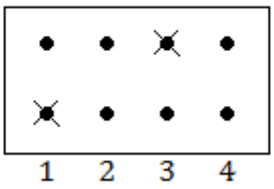
In paragraaf **3.3** hebben het we het probleem herschreven zodat alle restricties, op de geheeltalligheidsrestrictie na, zich in één matrix bevinden. Nu gaan we bekijken of zonder geheeltalligheidsrestrictie de oplossing van het probleem nog steeds geheeltallig is. Als dit zo is, kan het probleem efficiënt opgelost worden met een LP solver.

Totaal Unimodulair

In het vorige hoofdstuk hebben we aangetoond dat de restrictiematrix totaal unimodulair (TU) was. Hiermee konden we laten zien dat de oplossing van het LP geheeltallig was. Dat de matrix TU was wisten we door gebruik te maken van stelling **2.4** (opeenvolgendheid van de enen). In ons huidige probleem heeft de restrictiematrix deze eigenschap niet. Dit komt doordat de laatste kolommen van matrix C, horende bij variabele y, niet opeenvolgende verzamelingen van gecoverde punten hebben. Met het volgende tegenvoorbeeld laten we zien dat de restrictiematrix niet TU is.

Stel we hebben een paneel met punten $j = \{1,2,3,4\}$ aan zowel de boven- als onderkant. De maximale afstand is hier gelijk aan één. Daarnaast moeten we ook rekening houden met twee verboden punten. Aan de bovenkant is dat punt 3, en aan de onderkant punt 1. Figuur 3.3 laat het bijbehorende paneel zien.

Figuur 3.3



Hierbij horen de volgende matrices A, B en $A \cap B$.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad A \cap B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We kunnen van een TU matrix een kolom verwijderen zonder dat hierdoor de TU-eigenschap verandert. Dit is toegestaan aangezien elke submatrix ook TU is. Voor het overzicht laten we hier de nulkolommen van matrix C weg. Als matrix C TU is, dan is het dat ook zonder de weggelaten nulkolommen. Dit levert de volgende matrix C' op:

$$C' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Uit definitie **2.2** maken we op dat alle niet singuliere vierkante submatrices van een TU matrix unimodulair zijn. Dit betekent volgens definitie **2.1** dat alle vierkante submatrices een determinant moeten hebben gelijk aan {-1,0,1}. Als we één submatrix kunnen vinden die hier niet aan voldoet, hebben we bewezen dat matrix C' (en ook C) niet TU is.

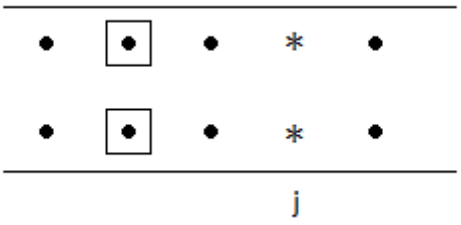
$$\begin{bmatrix} C'(3,8) & C'(3,7) & C'(3,5) \\ C'(6,8) & C'(6,7) & C'(6,5) \\ C'(8,8) & C'(8,7) & C'(8,5) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Aangezien de bovenstaande vierkante submatrix van C' een determinant gelijk aan 2 heeft, is matrix C niet TU. Dat de restrictiematrix niet TU is, betekent niet dat het bijbehorende probleem niet efficiënt op te lossen. In de volgende sectie gaan we op twee manieren laten zien dat dit wel kan. Als eerste laten we dit zien met een bewijs uit het ongerijmde.

Een bewijs van geheeltalligheid

Stel dat we de geheeltalligheidsrestrictie uit het ILP model relaxen en dat de optimale oplossing van het LP niet geheeltallig is. Oftewel $\exists j$ zodanig dat $0 < x_j < 1$. Een gedeelte van het bijbehorende cuppatroon laten we in figuur 3.4. Hierin zijn de door het LP geplaatste cups ($x_j = 1$) weergegeven als vierkanten. De niet geheeltallige variabelen geven we aan met een asterisk (*).

Figuur 3.4



Een gedeelte van het paneel. De vierkanten zijn geplaatste cups. De asterisken zijn niet geheeltallige variabelen op het punt j.

Als we van links (punt 1) naar rechts (punt m) het paneel bekijken, moet een soortgelijke situatie als in figuur 3.4 zich voordoen. Het hoeft echter niet zo te zijn dat de niet geheeltallige variabelen zich recht boven elkaar bevinden. Verder kan het zijn dat slechts één van de twee kanten een niet geheeltallige variabele heeft. Dit heeft geen invloed op de rest van het bewijs.

De enige reden dat het LP $x_j < 1$ toewijst in plaats van $x_j = 1$ is dat deze waarde van x_j genoeg is om alle punten in de buurt te coveren. Een punt j wordt gecoverd als

$$\sum_{i:|j-i| \leq max} x_i \geq 1$$

Echter is de waarde van x_j alleen niet genoeg om het punt j en de punten aan de rechterkant ervan ($j+1, j+2, \ldots$) te coveren. Dit betekent dat er een variabele $x_k > 0$ moet zijn zodanig dat ($k - j \leq max$). Gezamenlijk zouden deze twee variabelen wel de punten tussen j en k coveren;

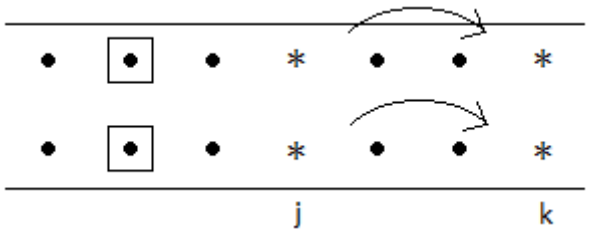
$$x_j + x_k \geq 1$$

We weten dat punt k binnen de maximale afstand van punt j ligt. De oplossing blijft toelaatbaar als we de volgende verandering aanbrengen:

$$x_k = 1, \qquad x_j = 0$$

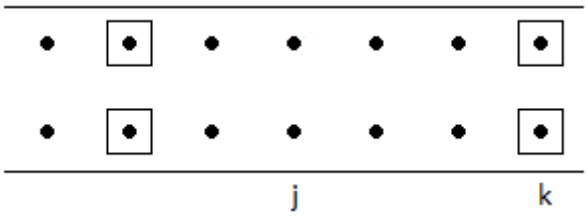
Hiermee coveren we nog steeds alle punten rond punt j, alleen zorgen we ervoor dat beide variabelen geheeltallig worden. Figuren 3.5 en 3.6 laten grafisch deze aanpassing zien.

Figuur 3.5



Omdat de variabelen bij het punt j niet de omringende punten geheel coveren, moeten er variabelen $x_k > 0$ aan de rechterkant te vinden zijn.

Figuur 3.6



We passen de variabelen aan zodat $x_j = 0$ en $x_k = 1$ .In deze situatie worden nog steeds alle punten rond j gecoverd, maar maken we de oplossing geheeltallig.

Deze procedure kunnen we voor alle $0 < x_j < 1$ herhalen. Dit maakt een geheeltallige oplossing van een oplossing die dat niet was. Omdat tijdens de procedure het aantal cups en balken niet toeneemt, is de verkregen oplossing optimaal. Dit is een tegenspraak met de aanname dat de optimale oplossing niet geheeltallig is.

Algemene voorwaarde voor geheeltalligheid

Naast de bovenstaande "ad hoc" redenering zijn er standaardmanieren om aan de restrictiematrix te zien, dat het probleem een geheeltallige oplossing heeft. In deze paragraaf doen we dit door gebruik te maken van een stelling van Hoffman en Schwartz[3]. Voordat we deze stelling kunnen toelichten en toepassen op ons probleem, moeten we enkele definities introduceren.

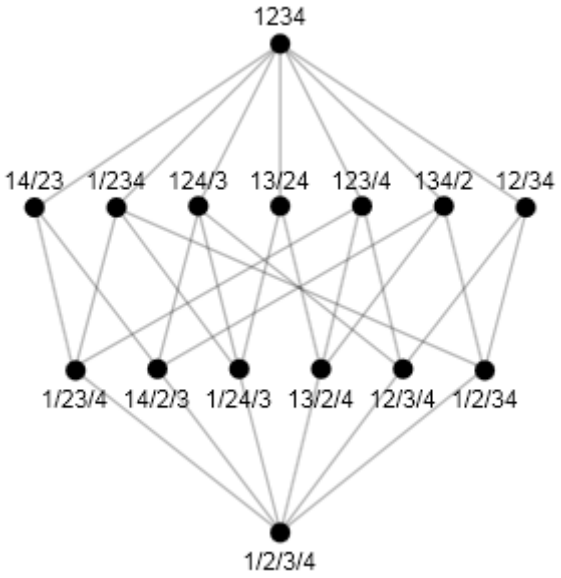
Definitie 3.1

*Stel dat we een relatie $\leq$ hebben op een verzameling L. Dan is $\leq$ een **partiële ordening** op L als voor alle a, b en $c \in L$ geldt:*

- $a \leq a$ *(L is reflexsief)*
- als $a \leq b$ en $b \leq a$ dan volgt $a = b$ (L is antisymmetrisch)
- als $a \leq b$ en $b \leq c$ dan volgt $a \leq c$ (L is transitief)

We zullen hier een voorbeeld van een partiële ordening geven. Stel we hebben een verzameling L die bestaat uit de 15 partities van de verzameling $\{1,2,3,4\}$. De verzameling L is partieel geordend door middel van de relatie verfijning. Een partitie is een verfijning ten opzichte van een andere partitie wanneer minstens één van de deelverzamelingen verder is opgedeeld. Een voorbeeld hiervan zijn de partities $a = \{12/3/4\}$ en $b = \{123/4\}$, waar de eerste een verfijning van de tweede is. Wanneer een tweetal partities gerelateerd zijn, noemen we ze **vergelijkbaar** ($a \leq b$). In figuur 3.7 wordt de partiële ordening weergegeven in een Hasse-diagram. De partities zijn hierin aangeduid als punten en de relaties tussen de partities zijn weergegeven als lijnen.

Figuur 3.7



De Hasse-diagram die de partiële ordening door middel van verfijning weergeeft.

Definitie 3.2

*Een verzameling met een partiële ordening is een **lattice** als elke twee elementen a en b een uniek infimum (grootste ondergrens) en supremum (kleinste bovengrens) hebben.*

- Infimum van a en b: $a \wedge b$
- Supremum van a en b: $a \vee b$

Als voorbeeld van een lattice gebruiken we dezelfde verzameling L waarvan we al hebben laten zien dat het een partiële ordening is. Deze verzameling is partieel geordend door middel van verfijning. L is een lattice als voor elk tweetal elementen uit L een uniek infimum en uniek supremum bestaat. We zullen eerst kort uitleggen wat een infimum en een supremum zijn. Stel dat we twee elementen a en b hebben waarvan we het infimum, de grootste ondergrens, willen bepalen. We kijken naar de verzameling $S \subseteq L$, $s \in S$ die er als volgt uit ziet:

$$S \coloneqq \{s | s \leq a, s \leq b\}$$

Het infimum van a en b is dan gelijk aan:

$$a \wedge b = \{\tilde{s} | \tilde{s} \geq s;\ \tilde{s}, \forall s \in S\}$$

Voor het supremum van a en b doen we ongeveer hetzelfde, alleen kijken we dan naar de verzameling van elementen die groter zijn dan a en b. Het kleinste element uit deze verzameling geeft ons het supremum.

We gaan nu aantonen dat elk tweetal elementen a en b een uniek infimum en uniek supremum heeft. We kijken eerst naar alle a en b die vergelijkbaar zijn. Stel dat bijvoorbeeld $a \leq b$, dan kunnen we zeggen dat:

$$a \vee b = b \quad \text{en} \quad a \wedge b = a,$$

waardoor duidelijk wordt dat een uniek infimum en uniek supremum bestaat wanneer a en b vergelijkbaar zijn.

Het kan ook zijn dat twee elementen a en b niet vergelijkbaar zijn. In figuur 3.7 is dat te zien als twee partities waartussen geen lijn getrokken is. Een voorbeeld hiervan zijn de partities $a = \{1/23/4\}$ en $b = \{12/3/4\}$. Deze zijn niet vergelijkbaar aangezien ze geen verfijning van elkaar zijn, maar hebben wel een uniek infimum en supremum. Voor het infimum kijken we naar de grootste partitie die een verfijning is van zowel a als b. In dit geval is dit $\{1/2/3/4\}$. Daarnaast kijken we naar de kleinste partitie waarvan zowel a als b een verfijning zijn. Hier is dat partitie $\{123/4\}$, oftewel:

$$a \vee b = \{123/4\} \quad \text{en} \quad a \wedge b = \{1/2/3/4\}$$

Deze suprema en infima bestaan voor alle a en b die niet vergelijkbaar zijn. Aangezien deze ook uniek zijn, hebben we bewezen dat verzameling L geordend door verfijning een lattice is.

Nu de definities van een partiële ordening en een lattice toegelicht zijn, vervolgen we met de stelling van Hoffman en Schwartz. Deze luidt als volgt:

Stel dat we een eindige partieel geordende verzameling L hebben. We nemen aan dat de twee functies " ∧ " en " ∨ " op $L \times L \to L$ bestaan zodanig dat $a \vee b = b \vee a \geq a, b$ en $a \wedge b = b \wedge a \leq a, b$, waarbij geldt dat zowel $a \vee b$ als $a \wedge b$ uniek zijn. Verder nemen we aan dat $a < b \Longrightarrow a \wedge b = a,\ a \vee b = b$. Daarnaast hebben we een eindige verzameling U en een afbeelding $f: L \to 2^U$ zodanig dat:

(1) $$a < b < c \in L,\quad u \in f(a)\ \ en\ \ u \in f(c)\ \implies u \in f(b).$$

Laat r een niet-negatieve geheeltallige functie gedefinieerd op L zijn zodanig dat voor alle $a, b \in L$:

(2) $$r(a \vee b) + r(a \wedge b) \leq r(a) + r(b).$$

Zij $S \subset U$ en χ de indicatorfunctie $\chi: 2^U \to \{0,1\}^U$, waarbij geldt dat element u van kolomvector $\chi(S)$ gelijk is aan 1 als $u \in S$. Oftewel, $\chi(S)$ wordt gedefinieerd als

$$\chi(S) = \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \begin{matrix} als\ u \notin S \\ \\ als\ u \in S \end{matrix}$$

Dan geldt voor alle $a, b \in L$:

(3) $$\chi\big(f(a \vee b)\big) + \chi\big(f(a \wedge b)\big) \geq\ \chi\big(f(a)\big) + \chi\big(f(b)\big).$$

<u>Stelling 3.1</u> (Hoffman en Schwartz) [3]

Stel dat aan (1),(2) en (3) is voldaan en dat c een niet-negatieve geheeltallige op U gedefinieerde functie is, dan geeft

$$min \sum_{a \in L} r(a)y(a)$$

$$s.t. \sum_{a|u \in f(a)} y(a) \geq c(u)$$

$$y(a) \geq 0,\ \ \forall a \in L$$

een geheeltallige vector.

We kunnen deze stelling gaan toepassen als we bewezen hebben dat aan de drie voorwaarden wordt voldaan. Voordat we dit gaan aantonen laten we eerst zien dat de verzameling L horende bij ons probleem een partiële orde heeft en dat het een lattice is.

Partiële orde

De verzameling L is bij ons gelijk aan de verzameling van kolommen uit de restrictiematrix C. Deze matrix hebben we gedefinieerd in **3.3** en ziet er als volgt uit:

$$C = \begin{bmatrix} A & 0 & A \cap B \\ 0 & B & A \cap B \end{bmatrix}$$

Deze matrix C gaan we opdelen in de drie submatrices:

$$C_1 = \begin{bmatrix} A \\ 0 \end{bmatrix}, \qquad C_2 = \begin{bmatrix} 0 \\ B \end{bmatrix} \qquad \text{en} \quad C_3 = \begin{bmatrix} A \cap B \\ A \cap B \end{bmatrix}.$$

Om de relaties tussen de kolommen duidelijk te maken, gaan we de kolommen van een andere naam voorzien. We stellen dat:

$$C_1 = [a_i], \qquad C_2 = [b_i] \qquad \text{en} \quad C_3 = [ab_i].$$

We gaan een relatie $\leq$ op de verzameling L definiëren zodanig dat in de matrices C_1, C_2 en C_3 alle kolommen reeds van klein naar groot zijn geordend. Oftewel wanneer deze matrices m kolommen hebben, dan geldt:

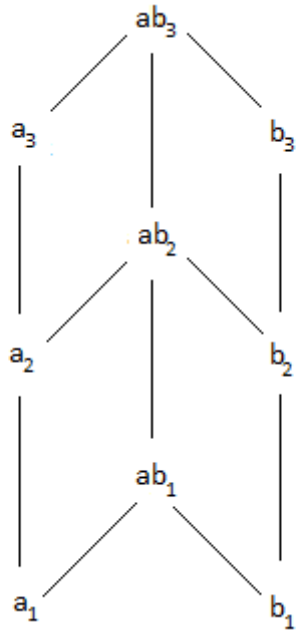
$$a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_m \ , \ b_1 \leq b_2 \leq b_3 \leq \cdots \leq b_m \ \text{ en } \ ab_1 \leq ab_2 \leq ab_3 \leq \cdots \leq ab_m$$

De kolommen a_i en b_j zijn onderling niet vergelijkbaar. Ze zijn allebei wel vergelijkbaar met de bijbehorende kolom uit submatrix C_3.

$$a_i \leq ab_i, \quad \forall i \quad \text{en} \quad b_j \leq ab_j, \quad \forall j$$

Hiermee wordt voldaan aan de drie genoemde eigenschappen van een partiële ordening. Daarnaast is de verzameling L van kolommen een eindige verzameling. Dit betekent dat L met relatie $\leq$ een eindige partieel geordende verzameling is. In figuur 3.8 laten we deze partiële ordening zien, waarbij $m = 3$.

Figuur 3.8



Hasse diagram van de partiële ordening van de kolommenverzameling L.

Lattice

We hebben aangetoond dat L een eindige partieel geordende verzameling is met relatie $\leq$. Van deze verzameling gaan we nu aantonen dat het een lattice is. Volgens definitie **3.2** betekent dit dat we voor elk tweetal elementen a en b moeten aantonen dat er een uniek infimum en supremum bestaat.

We gaan hier aantonen dat verzameling L een lattice is. We zullen twee gevallen onderscheiden. Eerst gaan we kijken naar het geval dat a en b vergelijkbaar zijn. Stel dat $a \leq b$. In dat geval geldt het volgende:

$$a \vee b = b \quad \text{en} \quad a \wedge b = a$$

Het is triviaal dat voor alle a en b die vergelijkbaar zijn, een uniek infimum en supremum bestaat.

Stel nu dat a en b niet vergelijkbaar zijn. In dat geval zijn er twee mogelijkheden.

- $a \in C_1$ en $b \in C_2$

In dat geval stellen we het infimum gelijk aan een nulkolom, oftewel:

$$a \wedge b = 0, \quad \text{waarbij geldt dat} \quad 0 \leq a, \ \forall a \in C$$

Om dit te bewerkstelligen moet we een nulkolom toevoegen aan de verzameling L. Dit kunnen we zonder probleem doen aangezien we hiermee geen invloed uitoefenen op de oplossing van het probleem. Daarnaast hebben we nog een andere situatie waarin twee kolommen niet vergelijkbaar zijn.

- $a \in C_1$ of C_2 en $b \in C_3$

We hebben bij de partiële ordening al aangeven dat deze twee kolommen vergelijkbaar zijn als we uit beide matrices dezelfde kolom pakken, bijvoorbeeld

$$a_i \leq ab_i$$

Ook zijn beide kolommen vergelijkbaar als $a_i \in C_1, C_2$ en $b_j \in C_3$ met $i < j$, aangezien dan geldt:

$$a_i \leq ab_i \leq ab_j$$

Echter als $i > j$ dan zijn a_i en ab_j niet vergelijkbaar. Het infimum voor dit tweetal is dan gelijk aan:

$$a_i \wedge ab_j = a_j$$

In figuur 3.8 is goed te zien dat dit de grootste ondergrens van a_i en ab_j is.

Hiermee is voor alle niet vergelijkbare tweetallen een uniek infimum gedefinieerd. Nu moeten we nog kijken naar het supremum voor alle niet vergelijkbare a en b . Er zijn drie situaties (waarbij de laatste twee bijna gelijk zijn) waarvoor het supremum al deel uit maakt van de verzameling L:

- $a_i \in C_1,\ b_i \in C_2 \longrightarrow a_i \vee b_i = ab_i$
- $a_i \in C_1,\ ab_j \in C_3$ met $i > j \longrightarrow a_i \vee ab_j = ab_i$
- $b_i \in C_2,\ ab_j \in C_3$ met $i > j \longrightarrow b_i \vee ab_j = ab_i$

Bij elk ander tweetal niet vergelijkbare kolommen is het supremum nog niet aanwezig in L. Al deze suprema gaan we definiëren en toevoegen aan de verzameling van kolommen. Dit kunnen we doen zolang we daarmee het probleem zelf niet veranderen. We zullen later beargumenteren wanneer dit het geval is. Hieronder laten we een voorbeeld van twee niet vergelijkbare kolommen a_i en b_j zien. Het supremum noemen we $a_i b_j$ en definiëren we op de volgende manier:

a_i	b_j	$a_i b_j$
$\begin{bmatrix}0\\1\\1\\1\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\1\\1\\1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\1\\1\\1\\1\\1\\0\end{bmatrix}$

Op deze manier kunnen we voor alle kolommen een uniek supremum toevoegen aan de verzameling. Echter kan het toevoegen van de suprema consequenties hebben voor het probleem. Om deze te begrijpen vertalen we de verzameling van kolommen naar het paneel waar we naar kijken. Kolom a_i staat voor de gecoverde punten als er een cup geplaatst wordt op punt i aan de bovenkant van het paneel. Bij kolom b_j kijken we naar de onderkant van het paneel en de punten die gecoverd worden wanneer er een cup op punt j wordt geplaatst. Het supremum van deze twee kolommen covert alle punten die ze allebei apart coveren.

We kunnen het supremum aan de verzameling van kolommen toevoegen, zolang het probleem daarbij niet verandert. Oftewel, het zou voor het LP model niet uit moeten maken of het enerzijds kolom a_i en b_j apart kiest, of dat het anderzijds het supremum ervan kiest. De kosten voor het kiezen van het supremum (de bijbehorende constante in de doelfunctie) zou gelijk moeten zijn aan die van a_i en b_j samen. Zolang we hiervoor zorgen kunnen we alle suprema toevoegen zonder dat de oplossing van ILP beïnvloed.

Ook voor deze nieuwe kolommen $a_i b_j$ moeten we unieke suprema en infima vaststellen. Deze definiëren we als volgt. Voor $i \leq l$ en $k \leq j$ geldt:

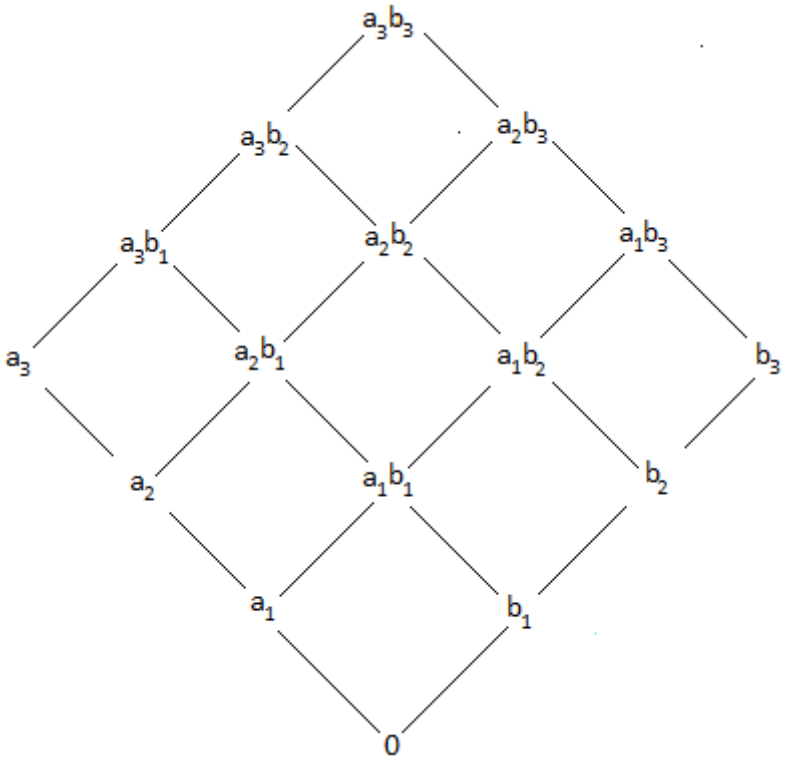
$$a_i b_j \vee a_l b_k = a_l b_j \quad \text{en} \quad a_i b_j \wedge a_l b_k = a_i b_k$$

Om het overzichtelijk te houden hernoemen we de kolommen ab_i naar $a_i b_i$. Bovenstaande infimum en supremum zijn ook dan van toepassing, aangezien voor $i \leq j \leq k$ geldt:

$$a_j b_j \vee a_i b_k = a_j b_k \quad \text{en} \quad a_j b_j \wedge a_i b_k = a_i b_j$$

We kunnen de verzameling L inclusief alle toegevoegde kolommen weergegeven in een Hasse diagram. Voor $m = 3$ (het bijbehorende paneel heeft in de discretisatie 3 kolommen) is deze diagram weergegeven in figuur 3.9.

Figuur 3.9



Hasse diagram van de uitgebreide verzameling L.

We hebben de verzameling L uitgebreid zodat alle elementen een uniek supremum en uniek infimum hebben. Hiermee hebben we aangetoond dat de uitgebreide verzameling L een lattice is.

Voorwaarden

We hebben aangetoond dat de verzameling L inclusief de toegevoegde kolommen een lattice is. Voordat we de stelling van Hoffman en Schwartz kunnen toepassen moet we nog laten zien dat aan voorwaarden (1), (2) en (3) wordt voldaan. Dit gaan we hier bewijzen.

Voorwaarde (1)

We hebben een eindige verzameling U en een afbeelding $f: L \to 2^U$ zodanig dat:

(1) $$a < b < c \in L, \quad u \in f(a) \quad en \quad u \in f(c) \quad \Longrightarrow u \in f(b).$$

De verzameling U is de verzameling van rijen van de restrictiematrix. Daarnaast maakt f een afbeelding van de kolommen op 2^U. Deze afbeelding kan gezien worden als het wel of niet kiezen van de rijen in de restrictiematrix. Als $u \in f(a)$ dan wordt rij u gekozen in kolom a.

Stel $a < b < c \in C_1$ en dat $u \in f(a)$ en $u \in f(c)$ maar $u \notin f(b)$. Door de ordening die we gekozen hebben weten we dat kolom b bij een punt hoort dat tussen a en c in ligt op het paneel. Verder weten we dat alle punten evenveel andere punten kunnen coveren, behalve als ze aan de rand van het paneel liggen. Aangezien $a < b < c$ kan punt b nooit minder punten coveren dan zowel punt a als punt c. Daarnaast is de verzameling van punten die gecoverd wordt een opeenvolgende verzameling. Als $j - 1$ en $j + 1$ gecoverd worden door een punt, dan moet j ook gecoverd worden.

Aangezien geldt dat $u \notin f(b)$ en $a < b$ moet dit betekenen dat rij u hoort bij een punt dat in de ordening kleiner is dan punt b en verder dan de maximale coverafstand weg ligt van punt b. Maar dan levert $u \in f(c)$ en $b < c$ een tegenspraak op, aangezien punt c er nog verder van af ligt.

Eenzelfde redenatie kan worden toegepast als $a < b < c \in C_2$, $a < b < c \in C_3$ of een combinatie ervan.

Voorwaarde (2)

Laat r een niet-negatieve geheeltallige functie gedefinieerd op L zijn zodanig dat voor alle $a, b \in L$:

(2) $$r(a \vee b) + r(a \wedge b) \leq r(a) + r(b).$$

Als a en b vergelijkbaar zijn is dit triviaal. Immers stel $a \leq b$, dan geldt:

$$a \vee b = b \quad \text{en} \quad a \wedge b = a,$$

en volgt

$$r(a \vee b) + r(a \wedge b) = r(a) + r(b).$$

Hetzelfde geldt voor $a \geq b$.

Als a en b niet vergelijkbaar zijn dan zijn er twee mogelijkheden:

- $a \wedge b = 0$

De variabele horende bij deze nulkolom zal door de LP solver niet gekozen worden in de oplossing, aangezien daar geen punten mee gecoverd worden. We kunnen de bijbehorende constante in de doelfunctie dan op nul zetten zonder de oplossing van het probleem te beïnvloeden. Oftewel, $r(a \wedge b) = 0$. Het kiezen van het supremum zou hier het plaatsen van twee balken betekenen. Dit geeft dat $r(a \vee b) = 2$. Het volgende geldt dan:

$$r(a) = 1, \quad r(b) = 1, \quad r(a \wedge b) = 0 \quad \text{en} \quad r(a \vee b) = 2,$$

en hieruit volgt:

$$r(a \vee b) + r(a \wedge b) = r(a) + r(b).$$

- $a \wedge b \neq 0$

Dit is zo als kolom a of b van de vorm $a_i b_j$ is. Stel dat $a = a_k$ en $b = a_i b_j$ met $k > i, j$ (anders waren de twee kolommen wel vergelijkbaar). Dan geldt dat:

$$a \vee b = a_k b_j \quad \text{en} \quad a \wedge b = a_i$$

De constante in de doelfunctie van een kolom van de vorm $a_i b_j$ is gelijk aan twee, aangezien het kiezen van deze kolom gelijk staat aan het kiezen van beide kolommen a_i en b_j apart. Dit geeft ons:

$$r(a) = 1, \quad r(b) = 2, \quad r(a \wedge b) = 1 \quad \text{en} \quad r(a \vee b) = 2,$$

en hieruit volgt wederom:

$$r(a \vee b) + r(a \wedge b) = r(a) + r(b).$$

Voorwaarde (3)

Zij $S \subset U$ en χ de indicatorfunctie $\chi: 2^U \to \{0,1\}^U$, waarbij geldt dat element u van kolomvector $\chi(S)$ gelijk is aan 1 als $u \in S$. Oftewel, $\chi(S)$ wordt gedefinieerd als

$$\chi(S) = \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \begin{matrix} als\ u \notin S \\ \\ als\ u \in S \end{matrix},$$

dan geldt voor alle $a, b \in L$:

(3) $$\chi\big(f(a \vee b)\big) + \chi\big(f(a \wedge b)\big) \geq \chi\big(f(a)\big) + \chi\big(f(b)\big).$$

Wanneer a en b vergelijkbaar zijn en $a \leq b$ geldt:

$$a \vee b = b \quad \text{en} \quad a \wedge b = a,$$

en volgt

$$\chi\big(f(a \vee b)\big) + \chi\big(f(a \wedge b)\big) = \chi\big(f(a)\big) + \chi\big(f(b)\big).$$

Hetzelfde geldt voor $a \geq b$.

Als a en b niet vergelijkbaar zijn er twee mogelijkheden:

- $a \wedge b = 0$

Dit is bijvoorbeeld zo als $a \in C_1, b \in C_2$. Het supremum is dan gedefinieerd als de som van beide kolommen. Dit betekent dat vergelijking (3) altijd een gelijkheid is.

- $a \wedge b \neq 0$

Dit is zo als a of b een kolom is van de vorm $a_i b_j$. Stel dat $a = a_k$ en $b = a_i b_j$. In dit geval geldt dat $k > i$, aangezien anders de kolommen vergelijkbaar zouden zijn. Het infimum en supremum zien er als volgt uit:

$$a \wedge b = a_i, \qquad a \vee b = a_k b_j$$

Wanneer we dit invullen in vergelijking (3) volgt:

$$\chi\left(f(a_k b_j)\right) + \chi\big(f(a_i)\big) \geq \chi\big(f(a_k)\big) + \chi\left(f(a_i b_j)\right).$$

Aangezien geldt dat:

$$a_k b_j = a_k + b_j \qquad \text{en} \qquad a_i b_j = a_i + b_j,$$

is vergelijking (3) wederom een gelijkheid. Dit is mogelijk voor alle $a \wedge b \neq 0$, waarmee we aan deze voorwaarde voldaan hebben.

Nu aan alle drie de voorwaarden voldaan is, kunnen we de stelling van Hoffman en Schwartz gaan toepassen op ons probleem.

Stelling Hoffman en Schwartz [3]

Stel dat aan (1),(2) en (3) is voldaan en c is een niet-negatieve geheeltallige op U gedefinieerde functie, dan geeft

$$min \sum_{a \in L} \quad r(a)y(a)$$

$$s.t. \sum_{a|u \in f(a)} \quad y(a) \geq c(u)$$

$$y(a) \geq 0, \quad \forall a \in L$$

een geheeltallige vector.

Als laatste moeten we een geheeltallige niet-negatieve functie c definiëren. In de stelling is te zien dat deze functie de rechterkant van de restricties vormt. In ons geval geldt dat $c = \mathbb{1}$. De conclusie van de stelling is dat dit LP een geheeltallige oplossing heeft. Echter hebben we een stelsel van vergelijkingen opgelost dat uitgebreider is dan die van ons cuppatroon probleem. We kunnen beargumenteren waarom dit geen gevolgen heeft voor de oplossing van het probleem. De enige kolommen die we hebben toegevoegd (naast een nulkolom die niet van invloed is) zijn de suprema van de vorm $a_i b_j$. Stel dat het LP ervoor heeft gekozen een supremum met de bijbehorende variabele te kiezen in de oplossing. Dan kunnen we deze variabele ook op nul zetten en in plaats daarvan de variabelen van a_i en b_j apart toevoegen. Aangezien geldt dat:

$$r(a_i) + r(b_j) = r(a_i b_j),$$

verandert hierdoor de waarde van het probleem niet.

3.5 Uitbreiding van de stabiliteitsvoorwaarde

In paragraaf **3.4** hebben we aangetoond dat het cuppatroon probleem met de gesimplificeerde stabiliteitsvoorwaarde een geheeltallige oplossing heeft. In hoofdstuk **4** gaan we kijken of dit ook het geval is met de originele stabiliteitsvoorwaarde zoals die geformuleerd is in paragraaf **1.5**. Voordat we dit doen gaan we beide voorwaarden in deze paragraaf uitbreiden. We zullen eerst met een alledaags voorbeeld duidelijk maken waarom de huidige definitie van stabiliteit volgens ons nog tekort schiet.

Stel dat we een stuk hout met de hand gaan zagen. Wanneer we het hout niet goed vastgehouden, zal het gaan meebewegen met het zaagblad. Deze instabiliteit kan leiden tot een lelijke snede. Dit voorkomen we door het stuk hout stevig vast te houden op de plek waar we aan het zagen zijn. Ditzelfde principe geldt bij het gebruik van de CNC-machine. We willen dat de plekken waar de machine een bewerking gaat uitvoeren aan het paneel, goed vastgezet worden door de vacuümcups.

Als we dit terugvertalen naar ons model betekent dit dat de punten waar een bewerking uitgevoerd wordt (**bewerkingspunten**) steviger vastgehouden moeten worden dan tot nu toe gebeurde. Dit kunnen we bereiken door de maximale afstand van dit soort punten tot de dichtstbijzijnde cup te verkleinen. Deze aangepaste maximale afstand bij bewerkingspunten noemen we max'. Merk op dat bewerkingspunten niet hetzelfde zijn als verboden punten. Bij de laatste kunnen we geen cup plaatsen vanwege de mogelijkheid dat de machine de vacuümcup raakt. Dit is bijvoorbeeld het geval bij een doorboring van het paneel.

Dynamisch programmeren

We gaan kijken wat het gevolg is van deze verandering bij het gebruik van dynamisch programmeren. De veranderingen vinden vooral plaats in de randwaarde condities. Stel S is de verzameling van bewerkingspunten. Dan geldt voor alle $j \in S$:

$$V_j(k,l) = \min\begin{Bmatrix} 1 + V_{j+1}(1,1), 1 + V_{j+1}(1,l+1), \\ 1 + V_{j+1}(k+1,1)\ ,0 + V_{j+1}(k+1,l+1) \end{Bmatrix}, \qquad k,l \in \{1,2,\dots,2 * max'\}$$

$$V_j(k,l) = \min\{1 + V_{j+1}(1,1), 1 + V_{j+1}(1,l+1)\}, \qquad l < k = 2 * max' + 1$$

$$V_j(k,l) = \min\left\{1 + V_{j+1}(1,1), 1 + V_{j+1}(k+1,1)\right\}, \qquad k < l = 2 * max' + 1$$

$$V_j(k,l) = 1 + V_{j+1}(1,1), \qquad l = k = 2 * max' + 1$$

Als $m \in S$, dan zijn de startwaarden als volgt:

$$V_m(k,l) = 0, \qquad k,l \in \{1,2,\dots,max'\}$$

$$V_m(k,l) = 1, \qquad k = max' + 1 \quad of \quad l = max' + 1$$

Als $1 \in S$, dan geldt:

$$V_1(max' + 1, max' + 1) = \min\begin{Bmatrix} 1 + V_2(1,1), 1 + V_2(1,max' + 2), \\ 1 + V_2(max' + 2,1)\ ,0 + V_2(max' + 2, max' + 2) \end{Bmatrix}$$

Voor alle andere punten, $j \notin S$, gelden de eerder in **3.2** weergegeven recursieve formules. Bovenstaande formules geven de situatie weer waarin zowel aan de boven- als de onderkant bewerkingspunten zijn. Het is ook mogelijk dat slechts aan één van beide kanten een bewerking plaatsvindt. In dat geval gelden de aangepaste randwaarde condities maar voor één kant.

LP model

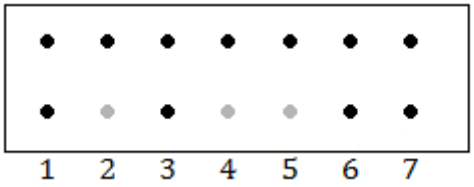
Het introduceren van bewerkingspunten heeft ook invloed op het LP model. In dit model hebben we restrictiematrix $A = [a_{ij}]$ gedefinieerd waarbij geldt dat $a_{ij} = 1$ als punt i gecoverd wordt door punt j. Dit is het geval wanneer punt j binnen een afstand van max ligt van punt i. Stel nu dat punt i een bewerkingspunt is. We willen dat dit punt beter vastgehouden wordt door de vacuümcups. Dit bereiken we door de maximale afstand tot de dichtstbijzijnde cup te verkleinen van max naar max'. Het bewerkingspunt i kan door deze aanpassing door minder punten gecoverd worden dan eerst het geval was. Dit betekent dat er in rij i van matrix A zich minder enen bevinden ($2 * max' + 1$ in plaats van $2 * max + 1$) dan in een rij die hoort bij een normaal punt. Dit laten we zien aan de hand van een voorbeeld.

Voorbeeld

We geven hier een voorbeeld om te laten zien wat er gebeurt als we bewerkingspunten toevoegen aan een paneel. Figuur 3.10 laat het paneel zien dat we als voorbeeld gaan gebruiken. Dit paneel heeft de volgende eigenschappen:

- Punten $\{1,2,3,4,5,6,7\}$
- Maximale afstand is 2
- Maximale afstand bij een bewerkingspunt is 1
- Punten $2,4$ en 5 aan de onderkant zijn bewerkingspunten

Figuur 3.10



Het paneel heeft aan de onderkant drie bewerkingspunten. Deze punten zijn te zien als grijze stippen.

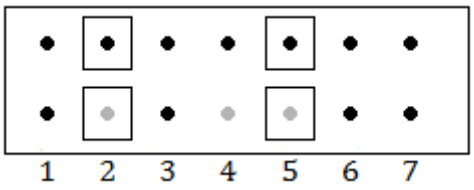
In de LP notatie hoort matrix A bij de bovenkant van het paneel en matrix B bij de onderkant.

$$A=\begin{bmatrix}1&1&1&0&0&0&0\\1&1&1&1&0&0&0\\1&1&1&1&1&0&0\\0&1&1&1&1&1&0\\0&0&1&1&1&1&1\\0&0&0&1&1&1&1\\0&0&0&0&1&1&1\end{bmatrix} \qquad B=\begin{bmatrix}1&1&1&0&0&0&0\\1&1&1&0&0&0&0\\1&1&1&1&1&0&0\\0&0&1&1&1&0&0\\0&0&0&1&1&1&0\\0&0&0&1&1&1&1\\0&0&0&0&1&1&1\end{bmatrix}$$

Het effect van de bewerkingspunten is te zien in de rijen 2,4 en 5 van matrix B.

Voor dit paneel gaan we een patroon van cups creëren. Dit patroon is weergegeven in figuur 3.11

Figuur 3.11



Het optimale patroon gebruikt twee balken.

In figuur 3.11 is te zien dat alle punten maximaal een afstand van twee verwijderd zijn van een cup. Daarnaast is de afstand van de drie bewerkingspunten tot de dichtstbijzijnde cup kleiner of gelijk aan één. Het aangepaste model houdt dus rekening met de punten waar het paneel beter vastgehouden moet worden.

We hebben in paragraaf **3.4** laten zien dat het cupprobleem voor deze panelen een geheeltallige oplossing heeft. Hier hadden we alleen nog geen bewerkingspunten waar we rekening mee moesten houden. Het is echter eenvoudig aan te tonen dat het cupprobleem met bewerkingspunten ook een geheeltallige oplossing heeft. Dit doen we door naar de procedure uit figuur 3.5 en 3.6 te kijken. Deze is nog steeds geldig als er bewerkingspunten op het paneel aanwezig zijn. Dit betekent dat ook met bewerkingspunten dit probleem efficiënt op te lossen is met een LP-solver.

Ook voor de smalle panelen uit hoofdstuk **2** kunnen we nog steeds een LP solver gebruiken. In dat hoofdstuk hebben we geconcludeerd dat dit kon vanwege stelling **2.4** die van toepassing was op de restrictiematrix in het LP model. Het toevoegen van bewerkingspunten zorgt ervoor dat er minder enen zijn in bepaalde rijen. Echter zijn deze enen nog steeds opeenvolgend, zodat stelling **2.4** nog steeds toepasbaar is. Hieruit volgt dat ook met het meenemen van bewerkingspunten in het model, de restrictiematrix totaal unimodulair blijft.

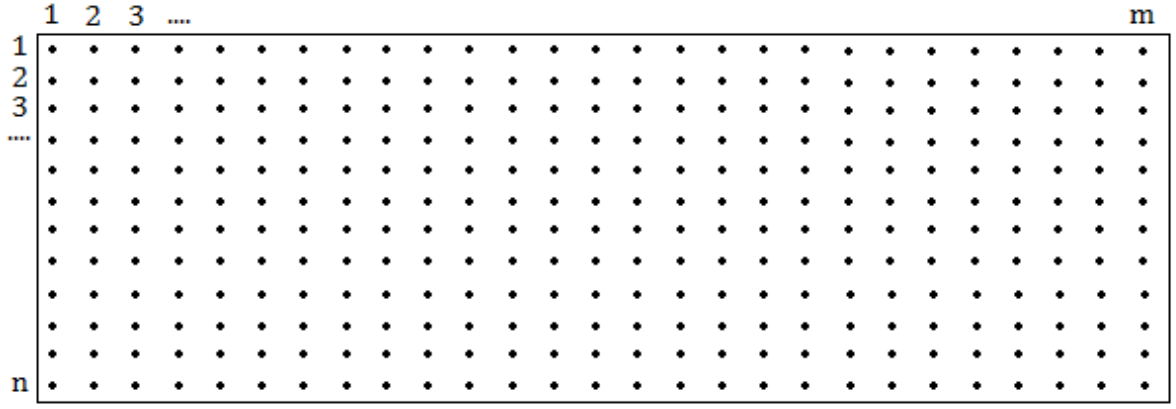
Conclusie

In dit hoofdstuk hebben we het model uitgebreid met panelen die stabiel gehouden moeten worden door twee cups boven elkaar te plaatsen. Het cuppatroon probleem voor deze panelen kan worden opgelost met behulp van dynamisch programmeren. De geheeltalligheid van de oplossing hebben we aangetoond door een niet geheeltallige oplossing te modificeren. Daarnaast hebben we ditzelfde ook bewezen met behulp van een stelling van Hoffman en Schwartz. Hieruit volgt dat het efficiënt oplossen van dit probleem mogelijk is door gebruik te maken van een LP solver. Ten slotte hebben we de stabiliteitsvoorwaarde uitgebreid door een extra een eigenschap aan bepaalde punten toe te voegen. We hebben laten zien dat deze aanpassing geen gevolgen heeft voor het efficiënt oplossen van het probleem.

4. Discretisatie van m × n punten

In paragraaf **1.5** hebben we laten zien op welke plekken van het paneel de vacuümcups geplaatst kunnen worden. De twee hoofdstukken daarna hebben we gekeken naar panelen die in deze discretisatie slechts één of twee rijen van punten gebruikten. In dit hoofdstuk gaan we kijken naar panelen waarvan de discretisatie uit n rijen punten bestaat. Figuur 4.1 laat zien hoe deze discretisatie eruit ziet.

Figuur 4.1



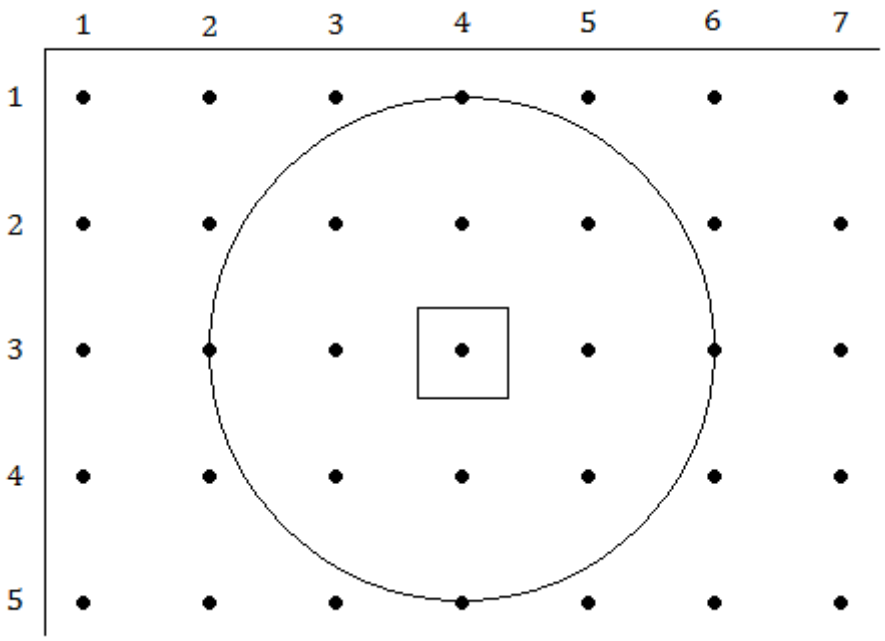
Rooster van punten waar een cup op het paneel geplaatst kan worden.

In het vorige hoofdstuk hebben we ervoor gekozen om te werken met een gesimplificeerde stabiliteitsvoorwaarde. Daarin hadden geplaatste cups alleen invloed op punten uit dezelfde rij in de discretisatie. In dit hoofdstuk gaan we weer werken met de originele stabiliteitsvoorwaarde. Het grote voordeel hiervan is dat dit een meer realistisch model oplevert. Het nadeel ervan is dat we hiermee ook de geheeltalligheid van de oplossing verliezen. Dit laten we zien in paragraaf **4.1**

4.1 Het ILP model

Doordat de discretisatie bestaat uit een rooster van punten, hoeven de cups niet meer in een vaste rij geplaatst te worden. Daarnaast gebruiken we niet meer de simplificatie van de stabiliteitsvoorwaarde. Als we dit vergelijken met hoofdstuk **3** dan zien we dat dit tot gevolg heeft dat de cups op een andere manier invloed hebben op de rest van het paneel. Zo zal nu elk ander punt dat binnen de maximale afstand van een cup ligt gecoverd worden. Dit geldt ook als dit punt zich in een andere rij bevindt. In figuur 4.2 laten we zien hoe dit in zijn werk gaat.

Figuur 4.2



Een uitvergroot gedeelte van het paneel. Het vierkant staat voor een geplaatste cup. De cirkel laat zien welke punten zich binnen de maximale afstand bevinden. In dit geval is deze afstand gelijk aan twee.

Nu een geplaatste cup ook invloed heeft op punten uit een andere rij, moeten we de afstand tussen twee punten gaan definiëren. De afstand tussen twee punten die naast of boven elkaar liggen is per definitie gelijk aan één. De overige afstanden kunnen berekend worden door de euclidische afstand te bepalen.

In de vorige hoofdstukken hebben we dynamisch programmeren gebruikt om een cuppatroon te berekenen. Bij het creëren van een optimale plaatsing van de cups, hoefden we in de recursieve formule alleen de afstand tot de volgende cup (variabele k) in de rij bij te houden. In de huidige situatie heeft plaatsing van een cup ook invloed op de omliggende rijen van punten. Het gebruik van dynamisch programmeren zou in principe nog steeds kunnen maar is wel minder efficiënt. Daarom gaan we in dit hoofdstuk het probleem alleen oplossen door gebruik te maken van een ILP formulering.

Dit zijn de eigenschappen van het ILP:

- Kolommen $j = \{1,2, \dots, m\}$ waar een balk geplaatst kan worden.
- Punten $h, i = \{1,2, \dots, m * n\}$ waar een cup geplaatst kan worden.
- Variabelen $y_j \in \{0,1\}$ die plaatsing van een balk weergeven. $y_j = 1$ als een balk op kolom j van het paneel geplaatst wordt.
- Variabelen $x_i \in \{0,1\}$ die plaatsing van een cup weergeven. $x_i = 1$ als een cup op punt i van het paneel geplaatst wordt.
- Restrictiematrix $A = [a_{hi}]$ met $a_{hi} \in \{0,1\}$. $a_{hi} = 1$ als punt h gecoverd kan worden door punt i. Oftewel, als punt i zich binnen de maximale afstand van punt h bevindt.
- Verzameling S_j van punten i die in kolom j liggen.

$$\min \sum_{j=1}^{m} y_j \qquad \text{(minimaliseer het aantal balken)} (0)$$

$$s.t. \sum_{i} a_{hi}\, x_i \geq 1, \qquad \forall h \qquad \text{(alle punten zijn gecoverd; stabiliteitsvoorwaarde)} (1)$$

$$\sum_{i \in S_j} x_i \leq 3, \qquad \forall j \qquad \text{(maximaal 3 cups per balk)} (2)$$

$$x_i \leq y_j, \qquad i \in S_j \qquad \text{(balk plaatsen als e een cup geplaatst is in de kolom)} (3)$$

$$x_i, \in \{0,1\}, \qquad \forall i \qquad \text{(elk punt krijgt wel of geen cup)} (4)$$

Ten opzichte van het LP uit **3.3** is een aantal dingen veranderd. In de doelfunctie (0) wordt nu alleen geminimaliseerd over alle y_j, waar we in **3.3** ook nog minimaliseerde over andere variabelen. Dit verschil met het eerder genoemde LP komt voort uit de andere discretisatie van het probleem. Doordat er slechts twee punten boven elkaar stonden, werd er verschil gemaakt tussen het plaatsen van een balk met daarop 1 of 2 cups. In het huidige model maken we dat onderscheid niet meer. Dit is voornamelijk een verschil in notatie en heeft geen invloed op het probleem zelf. Het ILP minimaliseert nog steeds het aantal balken dat gebruikt wordt in een cuppatroon.

Per balk kunnen maximaal drie cups het paneel vasthouden. Bij het LP uit **3.3** leverde dit geen extra restrictie op, omdat we daar panelen bekeken die slechts ruimte hadden voor twee cups. In de huidige situatie moet hier wel rekening mee gehouden. Dit wordt gemodelleerd in restrictie(2).

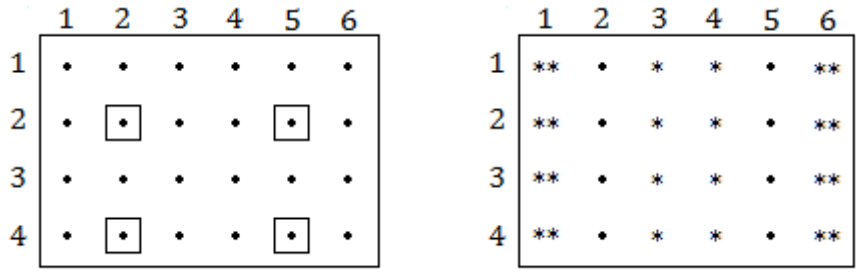
Geheeltalligheid

De laatste restrictie(4) van het ILP is om de geheeltalligheid van de oplossing te waarborgen. In de vorige hoofdstukken hebben we laten zien dat deze restrictie overbodig was. Zonder de restrictie zou de oplossing van het probleem nog steeds geheeltallig zijn. Doordat het model nu bestaat uit punten die niet alleen horizontaal maar ook verticaal op elkaar van invloed zijn, gaat dit niet meer op. Dit kunnen we laten zien aan de hand van een klein voorbeeld. We gaan twee keer hetzelfde probleem oplossen, alleen laten we bij één ervan de geheeltalligheidsrestrictie achterwege. Figuur 4.3 laat zien welke oplossingen dit oplevert.

Als voorbeeld nemen we een paneel met de volgende eigenschappen:

- Het aantal rijen is gelijk aan 4, oftewel $n = 4$.
- Het aantal kolommen is gelijk aan 6, oftewel $m = 6$.
- De maximale afstand is gelijk aan 2, oftewel $max = 2$.

Figuur 4.3



De panelen geven de patronen weer die twee verschillende solvers creëren. Links de geheeltallige oplossing van een ILP solver, waarin de vierkanten de cups voorstellen. Rechts is de oplossing van de LP solver te zien. De asterisken stellen variabelen voor die kleiner zijn dan 1. In dit geval $*=\frac{1}{7}$ *en* $**=\frac{2}{7}$.

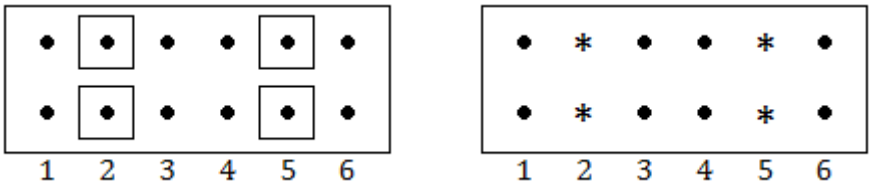
Figuur 4.3 laat zien dat bij de oplossing van het ILP 2 balken en 4 cups nodig zijn voor het patroon. Wanneer we een LP solver gebruiken, komen we tot het niet bruikbare aantal van $\frac{6}{7}$ balk dat nodig zou zijn om het hele paneel te coveren. De geheeltalligheidsrestrictie is dus weldegelijk noodzakelijk om tot een geheeltallige en bruikbare oplossing te komen.

Wanneer we in hoofdstuk **3** niet de gesimplificeerde stabiliteitsvoorwaarde hadden gebruikt, dan zouden we in dat model ook al geen geheeltalligheid meer gehad hebben. Dit laten we zien aan de hand van een tweede voorbeeld. Hierin laten we een LP solver voor twee gelijke panelen een cuppatroon berekenen. Het verschil tussen deze twee panelen is alleen de stabiliteitsvoorwaarde. Het ene paneel gebruikt de simplificatie uit hoofdstuk **3** en het andere paneel gebruikt de normale voorwaarde zoals die geformuleerd is in paragraaf **1.5**. De oplossingen die de LP solver geeft zijn te zien in figuur 4.4.

De panelen hebben beide de volgende eigenschappen:

- Het aantal rijen is gelijk aan 2, oftewel $n = 2$.
- Het aantal kolommen is gelijk aan 6, oftewel $m = 6$.
- De maximale afstand is gelijk aan 2, oftewel $max = 2$.

Figuur 4.4



*Het linker paneel voldoet aan de gesimplificeerde stabiliteitsvoorwaarde uit hoofdstuk **3**. Het rechter paneel gebruikt de huidige stabiliteitsvoorwaarde en heeft een niet geheeltallige oplossing. In dit geval geldt dat* $* = \frac{1}{2}$.

4.2 Een verwant probleem

Voordat we gaan kijken naar de efficiëntie van de oplosmethode, kijken we eerst naar een ander wiskundig probleem. In de grafentheorie wordt een graaf aangeduid met $G(V, E)$ waarbij V de verzameling van punten en E de verzameling van lijnen is. Hierbij geldt dat $(u, v) \in E$ een lijn voorstelt tussen de punten u en v.

Wanneer we een deelverzameling $U \subset V$ hebben zodat alle $v \in V$ aan minstens één van de volgende twee eigenschappen voldoet:

- $v \in U$
- Er bestaat een lijn $(u, v) \in E$ waarvoor geldt dat $u \in U$,

dan noemen we de verzameling U een **dominating set**. Een dominating set die zo min mogelijk punten bevat heet een **minimum dominating set** (MDS).

Ons cupprobleem kunnen we schrijven als een MDS probleem. De verzameling van punten V uit de graaf stellen we gelijk aan de verzameling van de punten waar het paneel uit bestaat. De verzameling E bestaat uit lijnen (u, v) voor alle punten u en v die zich binnen de maximale afstand van elkaar bevinden. Het oplossen van het MDS probleem geeft een verzameling punten die samen een dominating set vormen. Wanneer we op deze punten cups zouden plaatsen, hebben we een cuppatroon dat het gehele paneel stabiel houdt.

Speciaal geval: unit disk graphs

We gaan kijken naar het MDS probleem op een speciaal soort graaf. Deze graaf heet een **unit disk graph** (UDG). Bij een UDG ziet de verzameling E er als volgt uit:

$$(u, v) \in E \Leftrightarrow \| f(u) - f(v) \| \leq d,$$

met daarin afbeelding

$$f: V \longrightarrow \mathbb{R}^2,$$

en $\| . \|$ die de euclidische norm voorstelt. De verzameling E bestaat dus uit lijnen tussen alle punten die maximaal een afstand d van elkaar afliggen.

De naam van deze graaf komt van een bepaalde manier om hem weer te geven. We kunnen om alle $v \in V$ een cirkel tekenen met straal $\frac{1}{2}d$. Wanneer twee cirkels elkaar raken of snijden trekken we tussen de bijbehorende punten een lijn.

We hebben beargumenteerd dat ons cupprobleem vertaald kan worden naar een MDS probleem. Echter zal niet elke willekeurige graaf uit het MDS probleem een cuppatroon voortbrengen dat in ons model past. Vandaar dat we de UDG geïntroduceerd hebben. Door de afstand d gelijk te stellen aan de maximale afstand van een punt tot een cup (*max*) kan de oplossing van het MDS probleem op een UDG gezien worden als een cuppatroon dat een paneel stabiel houdt.

Complexiteit

Een oplossing van het bovenstaande probleem, levert ons ook een oplossing van het cuppatroon probleem. Van het MDS probleem op een UDG is bekend dat het NP-volledig is[4]. Voor dit probleem is geen efficiënte oplosmethode bekend. Het cuppatroon probleem is echter niet precies gelijk aan het MDS probleem op een UDG. Zo kunnen we een cuppatroon alleen toestaan wanneer deze gebruik maakt van maximaal acht balken met cups. Daarnaast is het niet toegestaan om meer dan drie cups per balk in het patroon te hebben. Het probleem is dus een specifiek geval van het MDS probleem op een UDG.

In sectie **2.3** hebben we aangetoond dat we een speciaal geval van een NP-volledig probleem (in dat geval het set cover probleem)toch efficiënt konden oplossen. Dit kwam door de bepaalde structuur die in de restrictiematrix aanwezig was. Of we dit probleem ook efficiënt kunnen oplossen is op dit moment een open vraag.

Stel dat het algoritme dat we gebruiken het probleem niet efficiënt kan oplossen. In dat geval zal de looptijd ervan exponentieel toenemen met de grootte van input. Wanneer we grote voorbeelden gaan bekijken, kan het voorkomen dat het niet lukt om snel een oplossing te verkrijgen. Het is echter wel de bedoeling dat het algoritme binnen enkele seconden een oplossing voor het probleem vindt. Door het algoritme te testen, komen we erachter of de gepresenteerde methode snel genoeg is om in de praktijk bruikbaar te zijn. Wanneer de methode te veel tijd kost, moeten we op een andere manier tot een patroon van cups komen. Een manier om dit te doen, is door de oplossing van het probleem te gaan benaderen.

Approximatie methode

Door de oplossing te benaderen met een approximatie methode kunnen we in een kortere tijd toch tot een oplossing van het probleem komen. Het idee achter het benaderen van de oplossing is dat er genoegen genomen wordt met een oplossing die niet optimaal is, maar wel binnen een bepaalde factor van de optimale oplossing ligt. Dit levert als voordeel op dat er een kortere looptijd nodig is dan bij een algoritme dat de optimale oplossing moet bepalen

Voor het MDS probleem op een UDG bestaat een benaderingsalgoritme dat elke factor boven de 1 kan benaderen [5]. Dit soort algoritmes worden ook wel PTAS (*polynomial time approximation scheme)* genoemd. De optimale oplossing kan benaderd worden tot elke factor $(1 + \varepsilon)$ met $\varepsilon > 0$. Oftewel als de optimale oplossing A is, dan zal het benaderingsalgoritme (bij een minimalisatie probleem) een oplossing geven die maximaal $(1 + \varepsilon)A$ groot is. Bij ons probleem werkt dit als volgt. Stel dat het minimaal aantal balken dat nodig is om een paneel vast te houden gelijk is aan 5. Het berekenen van een optimale oplossing kost echter dusdanig veel tijd dat we ook genoeg nemen met een patroon van cups dat gebruik maakt van 6 balken. In dat geval stellen we dat $\varepsilon = \frac{1}{5}$, aangezien geldt dat $1\frac{1}{5} * 5 = 6$.

Als blijkt dat onze optimale oplosmethode niet snel genoeg is, kunnen we besluiten deze benadering toe te passen. In hoofdstuk **5** testen we of dit noodzakelijk is.

4.3 Uitbreiding van het model

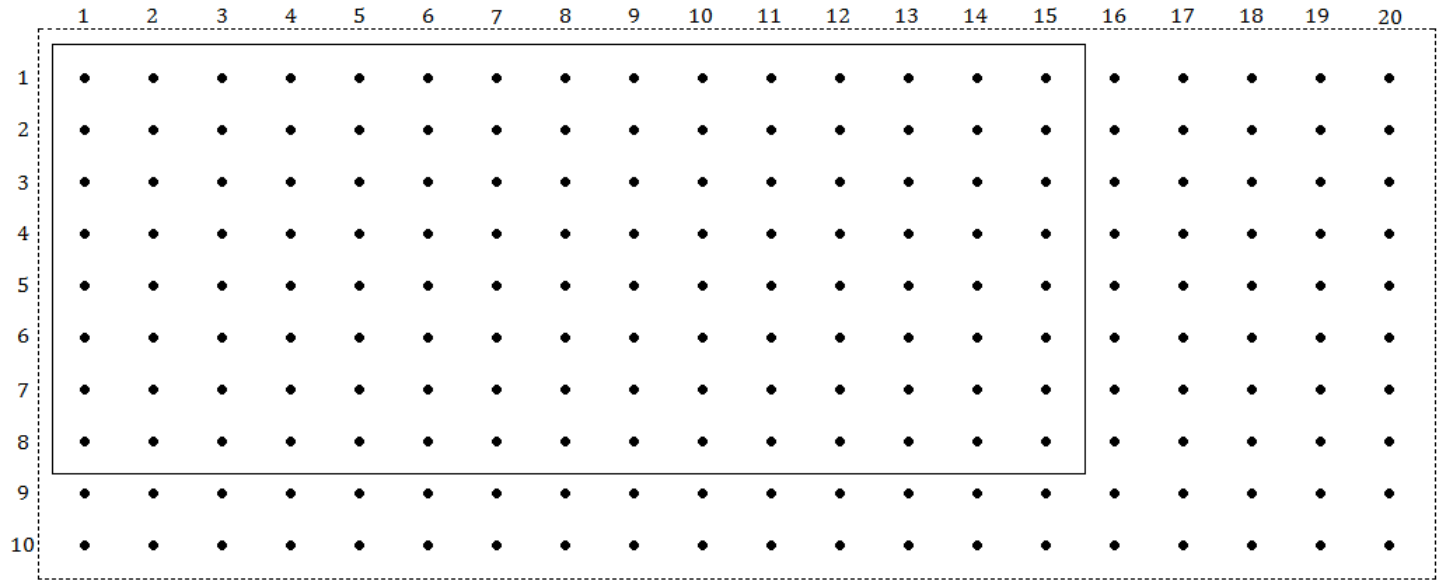
De computer horende bij de CNC-machine verkrijgt via een barcode alle benodigde informatie over het te bewerken paneel, zoals de afmetingen van het paneel en de bewerkingen die eraan uitgevoerd moeten worden. Het ILP model dat we in dit hoofdstuk gepresenteerd hebben houdt met al deze informatie rekening bij het creëren van een patroon. We hebben echter alleen nog maar gekeken naar de positionering van de cups die het paneel vasthouden. De cups die niet actief gebruikt worden in dit patroon kunnen ook van invloed zijn op het freesproces.

Het kan voorkomen dat er boringen gedaan moeten worden aan de zijkant van het paneel. Bij deze kopse boringen moet er rekening gehouden worden met de cups die niet het paneel ondersteunen. Om deze boringen uit te voeren heeft de machine aan de zijkant van het paneel genoeg ruimte nodig. De niet gebruikte cups mogen dan niet in de weg staan. Momenteel maken deze cups nog geen deel uit van het model. We kunnen alleen met deze cups rekening houden wanneer we de huidige discretisatie uitbreiden.

Discretisatie van het freesbed

We gaan de verzameling van punten in de discretisatie uitbreiden met punten van het gehele freesbed. Zo kunnen we ook bepalen waar de cups terecht komen die niet het paneel vasthouden. In figuur 4.5 is een discretisatie van het gehele freesbed te zien. In dit voorbeeld delen we het freesbed op in 10 rijen en 20 kolommen van punten. Op het freesbed is een paneel geplaatst waarvoor nog een patroon berekent moet worden.

Figuur 4.5



De discretisatie laat het hele freesbed zien. Het gestippelde vlak stelt het freesbed voor. De rechthoek is een te bewerken paneel dat zich op het freesbed bevindt. Op deze manier is het mogelijk om cups die niet het paneel vasthouden en plek op het freesbed toe te wijzen.

ILP model

Het aanpassen van de discretisatie heeft gevolgen voor het bijbehorende ILP model. Aangezien we nu kijken naar het gehele freesbed, moeten ook alle acht de balken met de bijbehorende drie cups een plek in het patroon krijgen. Dit levert het volgende ILP op:

$$\min \sum_{j=1}^{m} c_j y_j \qquad \text{(minimaliseer het aantal balken op het paneel)}$$

$$s.t. \sum_{i} a_{hi}\, x_i \geq 1, \qquad \forall h \qquad \text{(alle punten op het paneel zijn gecoverd)}$$

$$\sum_{j=1}^{m} y_j = 8 \qquad \text{(8 balken worden geplaatst)}$$

$$\sum_{i \in S_j} x_i = 3, \qquad \forall j \,|y_j = 1 \qquad \text{(3 cups per balk)}$$

$$x_i \leq y_j, \qquad i \in S_j \qquad \text{(balk plaatsen als er een cup geplaatst is in de kolom)}$$

$$x_i, \in \{0,1\}, \qquad \forall i \qquad \text{(elk punt krijgt wel of geen cup)}$$

Hierin zijn c_j de kosten voor het plaatsen van een balk bij kolom j. Aangezien wij het aantal balken dat gebruikt wordt om het paneel te ondersteunen willen minimaliseren ziet c_j er als volgt uit. Stel dat het paneel uit m_1 kolommen bestaat, dan geldt:

$$c_j = 1, \qquad \forall j \leq m_1$$

$$c_j = 0, \qquad \forall j > m_1$$

Dit zou voor het paneel in figuur 4.5 betekenen dat

$$c_j = 1, \quad j \in \{1,2,3,\dots,15\}$$

$$c_j = 0, \quad j \in \{16,\dots,20\}$$

De oplossing van het ILP model geeft nu een cuppatroon waar alle 24 cups deel van uitmaken. Deze oplossing houdt alleen nog geen rekening met eventuele kopse boringen. Met een kleine uitbreiding van de restricties is dit wel het geval. We gaan de punten naast een kopse boring markeren als verboden punten. Voor het ILP levert dit per verboden punt één extra restrictie op. De variabele horende bij het verboden punt krijgt hierin de waarde nul toegewezen. Oftewel, stel punt j ligt naast een plek waar een kopse boring plaats vindt, warbij het uitvoeren van deze boring niet mogelijk zou zijn als er op punt j een cup gepositioneerd staat. Om te voorkomen dat het algoritme hier toch een cup plaatst, voegen we de restrictie $x_j = 0$ toe aan het ILP. Op deze manier voorkomen we dat er cups in de weg staan bij het uitvoeren van kopse boringen.

Conclusie

In dit hoofdstuk hebben we een $m \times n$ discretisatie van het paneel gebruikt. Het voordeel hiervan is dat het een realistisch model geeft. Het nadeel is dat we de geheeltalligheid van de oplossing hiermee verliezen. We hebben laten zien dat het probleem een speciaal geval is van het dominating set probleem. Of het model efficiënt opgelost kan worden is een open vraag. Wanneer de input dusdanig groot is dat het algoritme te traag wordt om in de praktijk te gebruiken, moeten we teruggrijpen op een benaderingsalgoritme. Of dit nodig is zal blijken uit de tests die we in hoofdstuk **5** gaan uitvoeren. Daarnaast hebben we de discretisatie uitgebreid om zo het gehele freesbed te kunnen modelleren. Hiermee is het mogelijk om ook de cups die niet actief deel uitmaken van het cuppatroon een positie op het freesbed toe te wijzen. Het grote voordeel hiervan is dat we kunnen voorkomen dat er cups in de weg staan als er kopse boringen uitgevoerd moeten worden.

5. Resultaten

In dit hoofdstuk laten we zien welke patronen het gepresenteerde model voor een aantal verschillende panelen creëert. Verder testen we of het algoritme snel genoeg is om gebruikt te worden in de praktijk.

Oplosmethode

In hoofdstuk **4** hebben we een discretisatie van het paneel gepresenteerd waarbij het mogelijk is om drie cups op één balk te plaatsen. Dit model hebben we daarnaast nog uitgebreid door ook rekening te houden met de cups die niet het paneel vasthouden. Dit levert een stelsel vergelijkingen op waarvan de oplossing een plaatsing van alle 24 cups weergeeft.

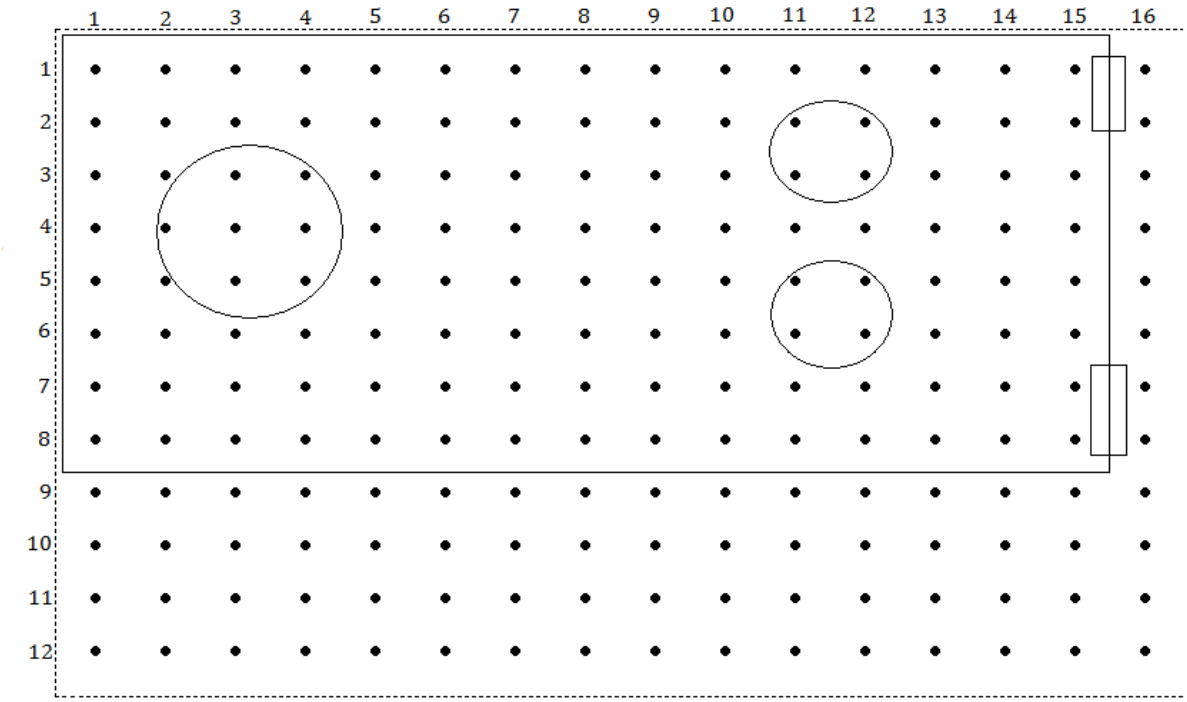
Voor het oplossen van het stelsel vergelijkingen gebruiken we het programma IBM ILOG CPLEX Optimization Studio (CPLEX). De vraag is we met dit programma snel genoeg tot een oplossing van het model kunnen komen. Dit gaan we testen door voor een aantal panelen een cuppatroon te berekenen en bij te houden hoelang het algoritme daarover doet.

De CNC-machine die door Kast op Maat gebruikt wordt heeft een freesbed dat ongeveer 120 cm breed en 280 cm lang is. Voordat we dit freesbed kunnen discretiseren moeten we beslissen hoever de punten in de discretisatie uit elkaar komen te staan. We hebben gekozen om de punten op een afstand van 10 cm van elkaar te zetten. Aangezien de cups zelf ongeveer 10 bij 10 cm groot zijn, zou het dichter op elkaar zetten van de punten problemen kunnen opleveren wanneer twee cups direct naast elkaar geplaatst worden. Hieruit volgt dat de discretisatie van het freesbed bestaat uit 12 rijen en 28 kolommen. We stellen dat een paneel stabiel gehouden wordt, als elk punt maximaal 20 cm van een cup verwijderd is.

Het paneel dat we eerst gaan bekijken heeft een lengte van 150 cm en een breedte van 80 cm. Bij het plaatsen van de cups moet er met een aantal bewerkingen rekening gehouden worden. In dit geval moeten er drie cirkels uit het paneel gezaagd worden. Daarnaast worden er aan de zijkant van het paneel twee kopse boringen uitgevoerd. Beide bewerkingen zorgen voor de nodige verboden punten. In figuur 5.1 is te zien op welke plekken van het paneel deze bewerkingen van invloed zijn. Welke verboden punten dit oplevert is te zien in figuur 5.2.

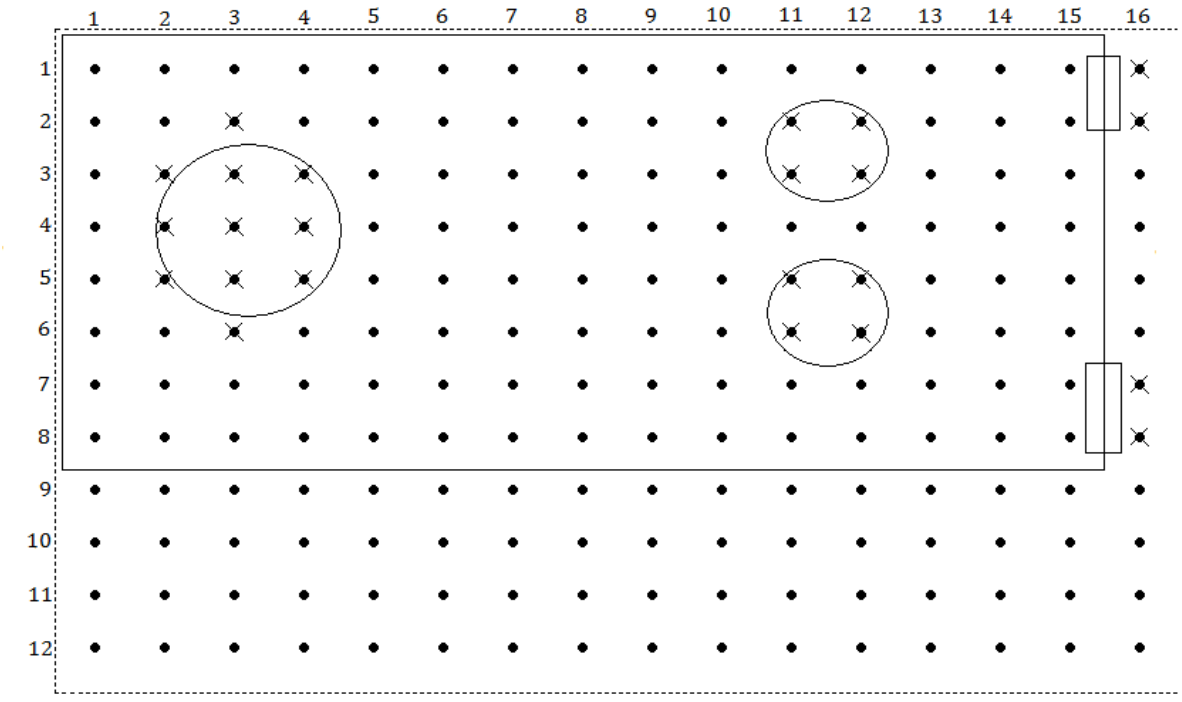
Beide figuren 5.1 en 5.2 laten slechts het gedeelte van het freesbed zien waar het paneel zich bevindt. Dit is gedaan om de figuren overzichtelijk te houden. De discretisatie uit figuur 5.2 zetten we om in het stelsel van vergelijkingen uit hoofdstuk **5**. Met behulp van CPLEX gaan we dit stelsel oplossen. Dit levert ons een patroon van cups op zoals te zien is in figuur 5.3. In dit figuur is wel het hele freesbed zichtbaar, waarin te zien is dat er zes balken nodig zijn om het paneel stabiel te houden. De andere twee balken met bijbehorende cups staan zo gepositioneerd, dat ze niet geraakt worden door de freesmachine bij het uitvoeren van de kopse boringen. De tijd die het algoritme nodig had om tot deze oplossing te komen was 3.09 seconde.

Figuur 5.1

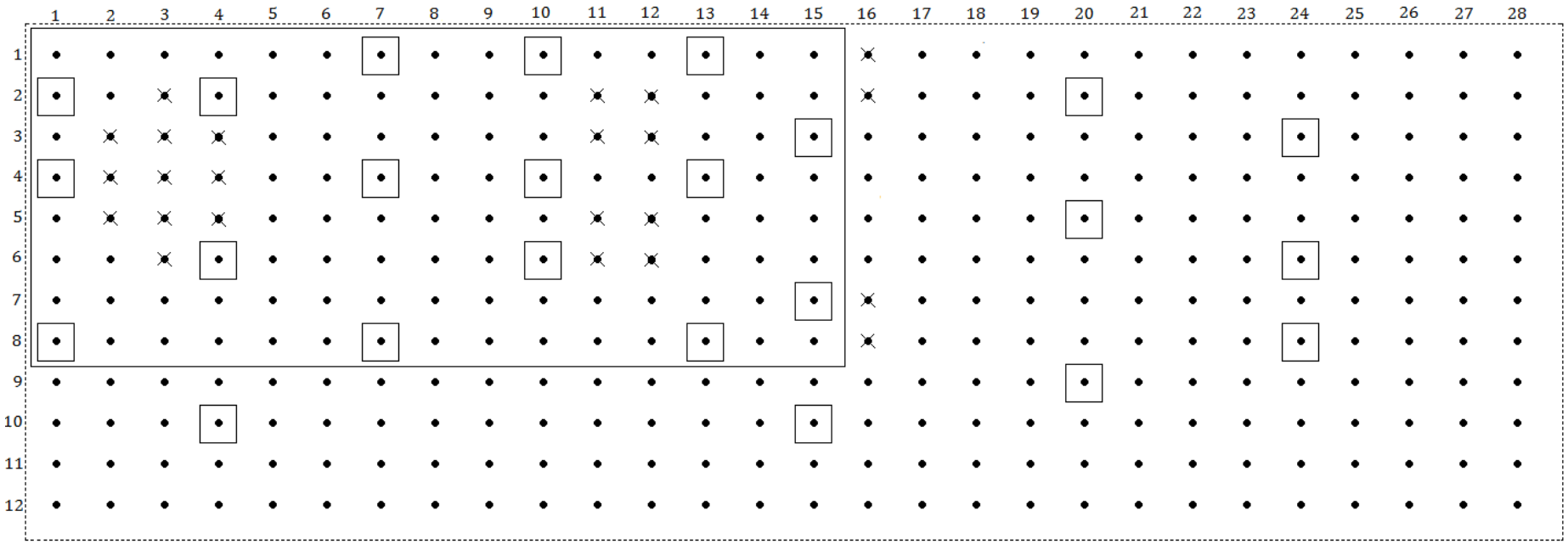


Uit het paneel moeten drie cirkels gefreesd worden. De punten in deze cirkels zijn verboden punten. De punten die net buiten de cirkels liggen zijn ook verboden, aangezien plaatsing van een cup ook hier problemen op kan leveren. De twee rechthoeken aan de rechterkant van het paneel staan voor de twee kopse boringen die daar plaatsvinden. In figuur 5.2 zijn de verboden punten aangekruist.

Figuur 5.2



Figuur 5.3

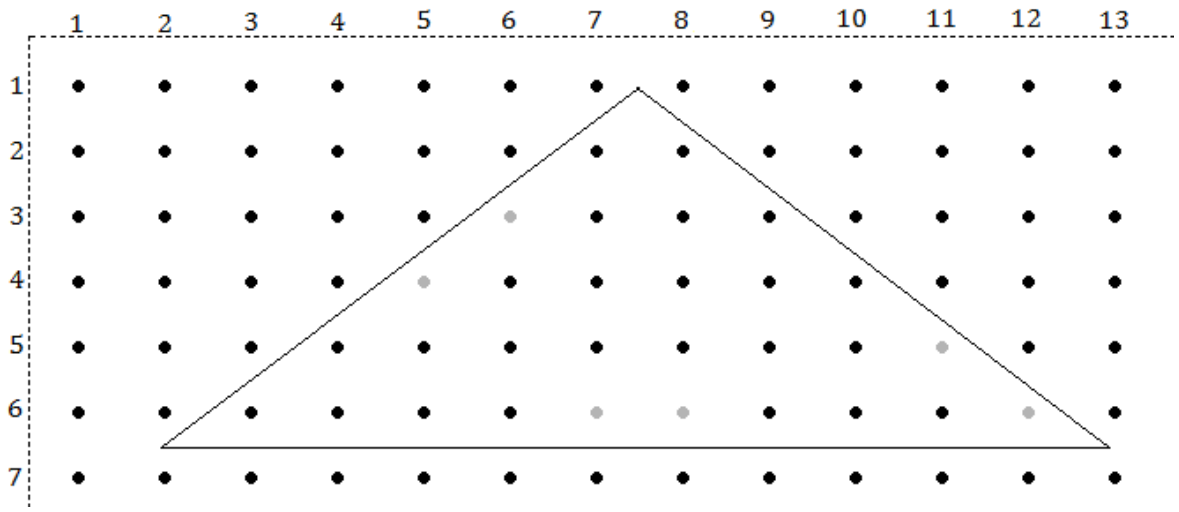

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
1
2
3
4
5
6
7
8
9
10
11
12

5.1 Niet rechthoekige panelen

Het voordeel van de huidige discretisatie is dat we niet gebonden zijn aan een bepaalde vorm van het paneel. Naast rechthoekige panelen moet Kast op Maat ook panelen met een andere vorm bewerken. Enkele van deze panelen gaan we nu bekijken.

In dit voorbeeld gaan we een patroon creëren voor een driehoekig paneel. Aan dit paneel moeten alleen oppervlakkige bewerkingen uitgevoerd worden. Hierdoor zijn er geen verboden punten aanwezig. Er bevinden zich wel zes bewerkingspunten op het paneel welke weergegeven zijn als een grijze stip. Deze bewerkingspunten willen we beter vasthouden dan de normale punten. Vandaar dat bij elk van deze bewerkingspunten de maximale afstand tot de dichtstbijzijnde cup gelijk is aan 10 cm, in plaats van de 20 cm bij de rest van de punten. Net als in figuren 5.1 en 5.2 laten we in de komende figuren alleen het gedeelte van het freesbed zien waar het paneel op ligt. Figuur 5.4 laat zien hoe het te bewerken paneel eruit ziet.

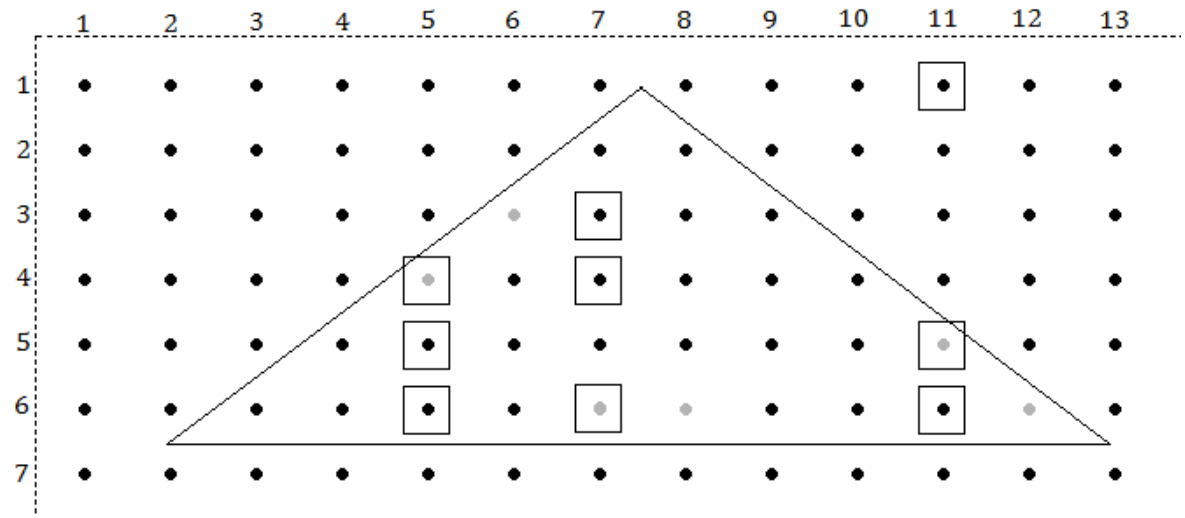
Figuur 5.4



Het te bewerken paneel heeft een driehoekige vorm. De grijze punten zijn de bewerkingspunten.

Figuur 5.5 laat zien dat dit paneel met drie balken vastgehouden kan worden. De zes bewerkingspunten hebben allemaal een cup op maximaal 10 cm afstand staan. Daarnaast is elk ander punt op het paneel maximaal 20 cm verwijderd van een cup, wat betekent dat aan de stabiliteitsvoorwaarde is voldaan. Het algoritme had in dit geval 0.39 seconde nodig om tot deze oplossing te komen.

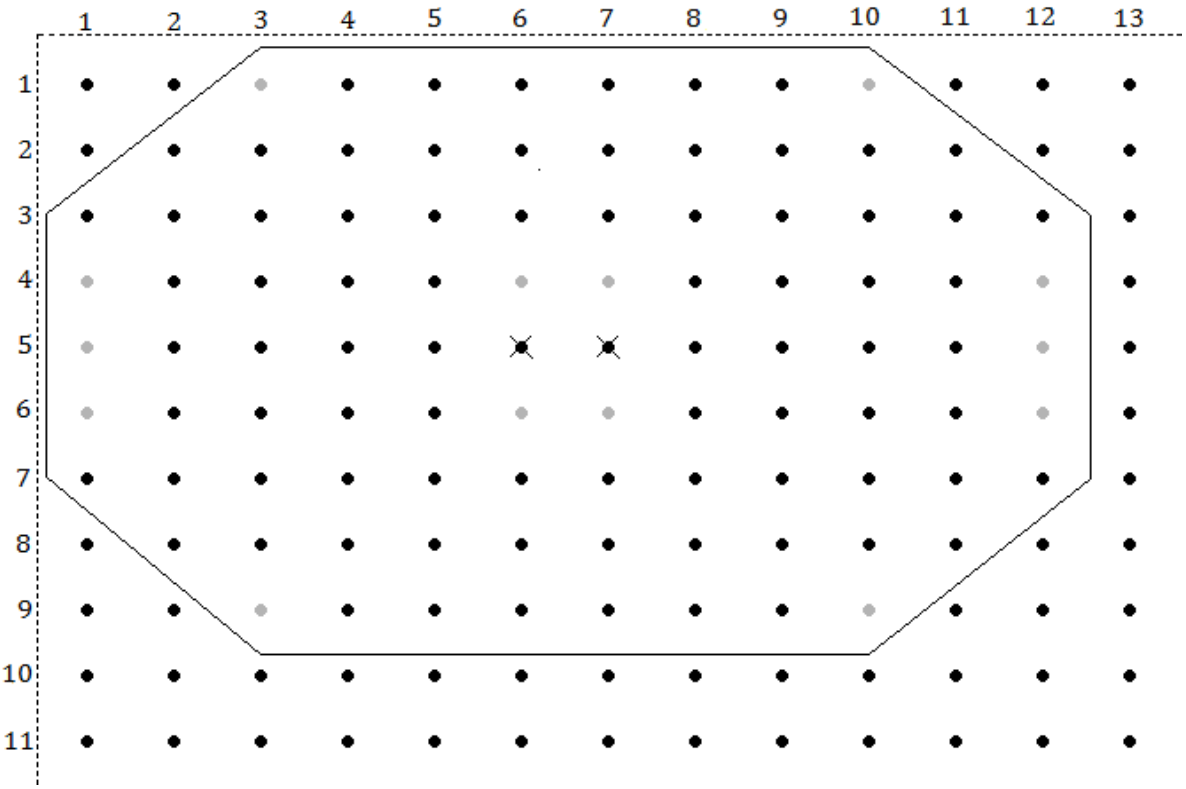
Figuur 5.5



Het optimale cuppatroon van dit paneel gebruikt drie balken.

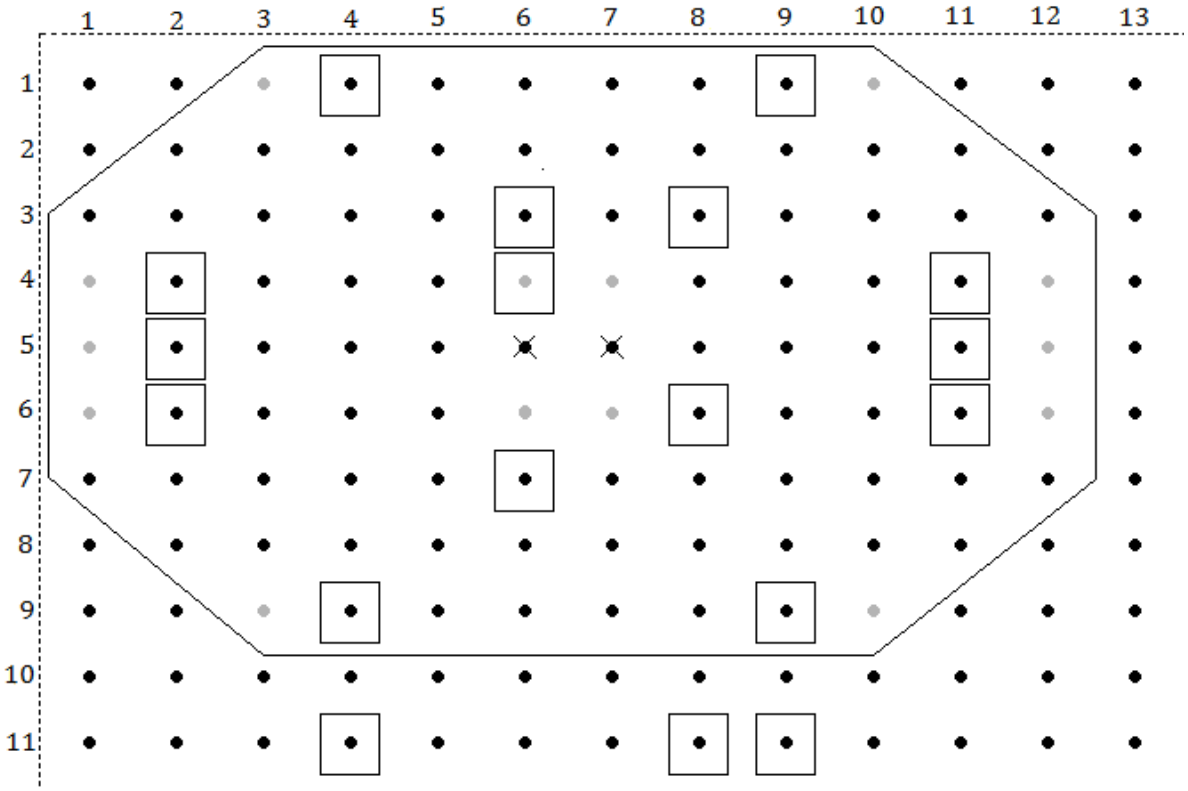
Naast driehoeken moeten er ook andere veelhoekige panelen bewerkt worden. In het volgende voorbeeld laten we een achthoekig paneel zien. Vanwege een doorboring bevinden zich midden op het paneel twee verboden punten. Daarnaast wordt er bij een aantal punten oppervlakkige bewerkingen uitgevoerd. In figuur 5.6 is te zien waar de verboden en bewerkingspunten op het paneel liggen. Figuur 5.7 laat zien welk cuppatroon het algoritme creëert. Voor het overzicht laten we wederom alleen het gedeelte van freesbed zien waar het paneel op ligt.

Figuur 5.6



Het achthoekige paneel bevat zowel verboden als bewerkingspunten.

Figuur 5.7



Om het achthoekige paneel stabiel te houden zijn er zes balken nodig.

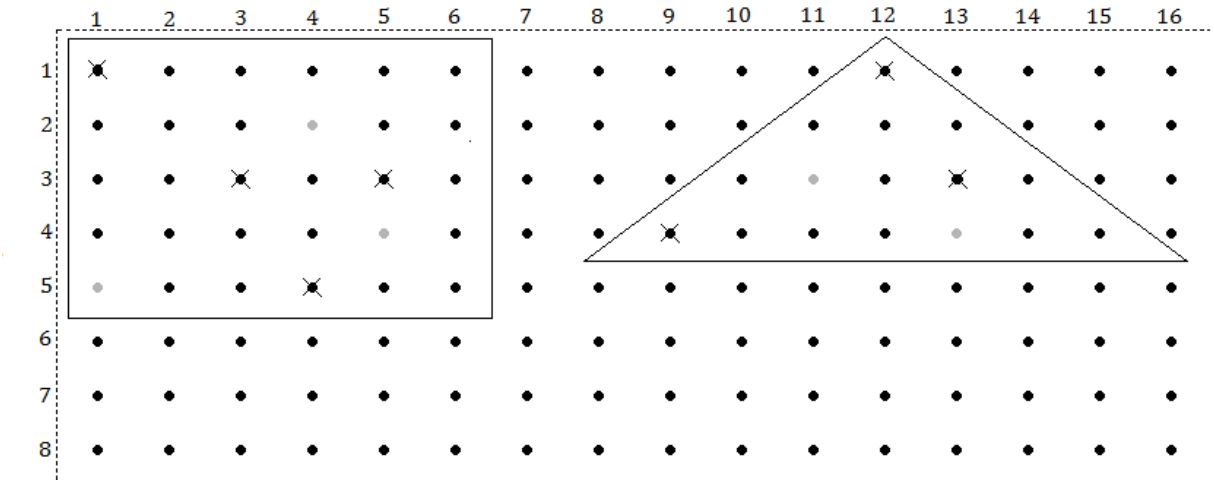
Figuur 5.7 laat zien dat het optimale patroon gebruik maakt van zes balken. De positionering van de overige twee balken is in dit geval niet van groot belang, aangezien er geen kopse boringen uitgevoerd worden. Het algoritme had 0.44 seconde nodig om dit patroon van cups te creëren.

Meerdere panelen

In hoofdstuk **2** hebben we beargumenteerd waarom een cuppatroon dat het minste aantal balken gebruikt optimaal is. Door het aantal balken per paneel te minimaliseren kan het freesbed beter benut worden door meerdere panelen tegelijk erop te leggen. Op deze manier kan de CNC-machine direct doorgaan met het frezen van het volgende paneel wanneer het klaar is met het eerste paneel. In het volgende voorbeeld gaan we kijken hoe we meerdere panelen op het freesbed kunnen leggen.

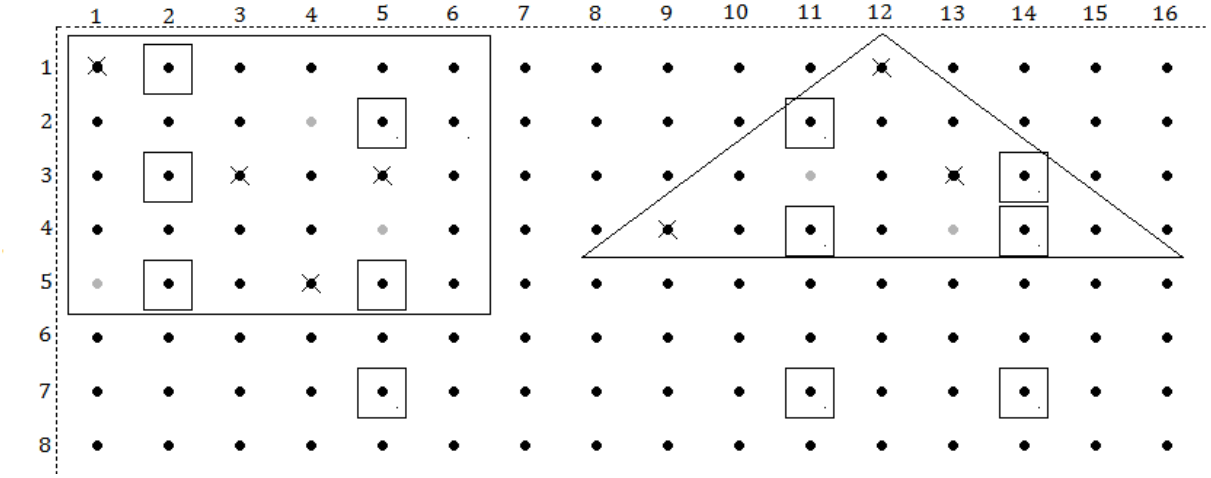
Figuur 5.8 laat de panelen zien die we gezamenlijk op het freesbed willen leggen. Deze panelen bevatten een aantal verboden en bewerkingspunten. Figuur 5.9 laat zien wat de oplossing van het algoritme is.

Figuur 5.8



De twee te bewerken panelen bevatten beide verboden en bewerkingspunten. Door ze gezamenlijk op het freesbed te leggen, kan de machine ze direct na elkaar frezen.

Figuur 5.9



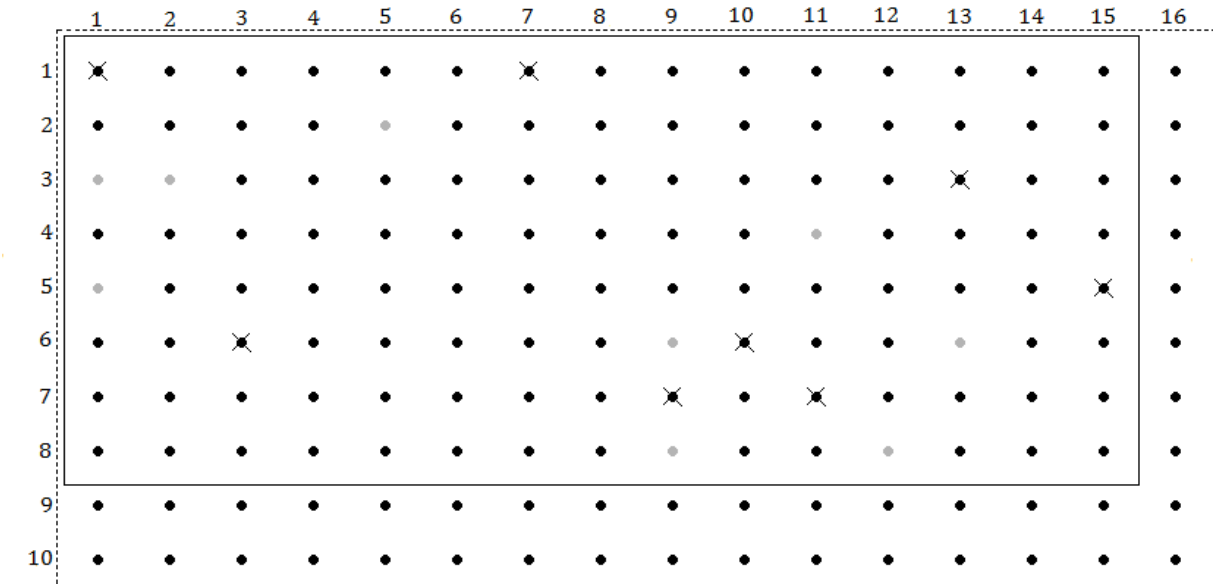
Beide panelen zijn stabiel te houden met twee balken. Door het aantal balken per paneel te minimaliseren, kunnen er meerdere panelen tegelijk bewerkt worden door de machine.

In figuur 5.9 is te zien wat het voordeel is van het minimaliseren van het aantal gebruikte balken in een cuppatroon. Hierdoor kunnen we meerdere panelen tegelijkertijd op het freesbed stabiel houden. De freesmachine kan dan meteen doorgaan met het frezen van het volgende paneel. Het algoritme deed er 0.24 seconde over om dit patroon te creëren.

Verschil in looptijd

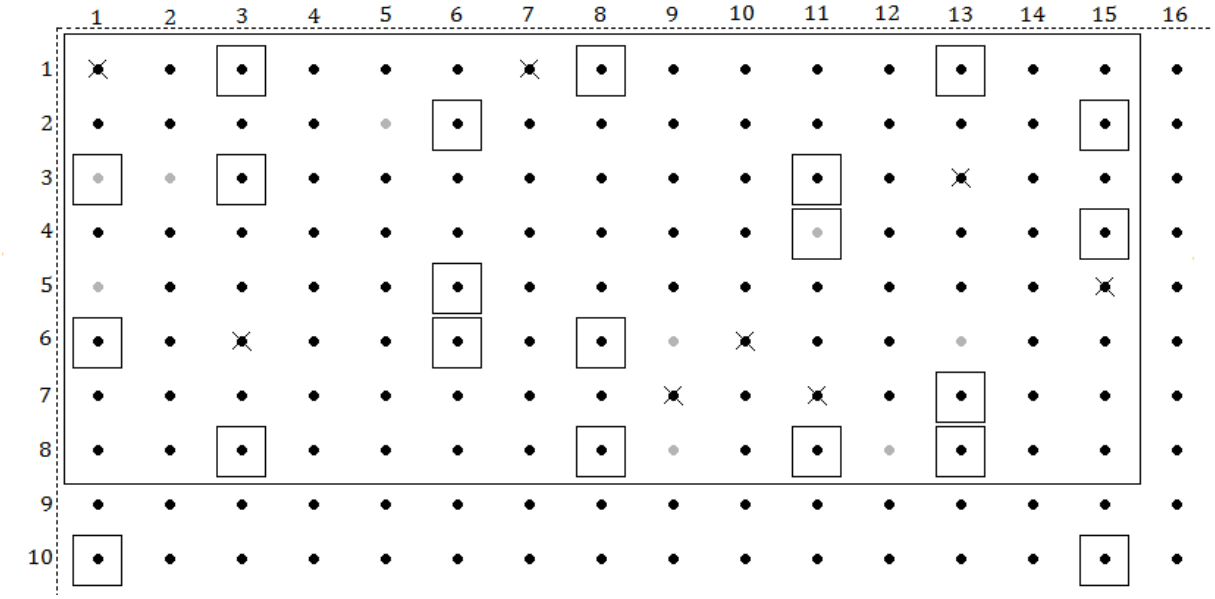
De behandelde voorbeelden in dit hoofdstuk lijken aan te tonen dat het algoritme snel genoeg tot een oplossing komt. Dat dit niet altijd het geval is laten we zien aan de hand van het volgende voorbeeld. We nemen een paneel met dezelfde afmeting als het paneel uit figuur 5.3, alleen veranderen we de verzamelingen van verboden en bewerkingspunten. Figuur 5.10 laat zien welk paneel dit oplevert. In figuur 5.11 is te zien welk patroon het algoritme voor het paneel creëert.

Figuur 5.10



Het paneel heeft dezelfde afmetingen als een het eerste voorbeeld uit dit hoofdstuk. Wat is veranderd zijn de verboden en bewerkingspunten.

Figuur 5.11



Het algoritme heeft 32.26 seconden nodig om tot deze oplossing te komen. Dit optimale patroon gebruikt zeven balken om het paneel stabiel te houden.

Het kostte het algoritme een stuk meer tijd om de oplossing uit figuur 5.11 te berekenen, namelijk 32.26 sec. Dit is meer dan 10 keer zo lang als het paneel met dezelfde afmetingen uit figuur 5.3. Een andere verzameling van verboden en bewerkingspunten kan dus een heleboel invloed hebben op de looptijd. In de volgende paragraaf gaan we kijken welke eigenschappen van een paneel er precies voor zorgen dat de looptijd van het algoritme veel toeneemt.

5.2 Looptijd

Naast dat het algoritme een patroon moet creëren dat het paneel stabiel houdt, is het van belang dat het algoritme niet te lang moet rekenen aan een oplossing. Het algoritme is niet bruikbaar in de praktijk wanneer de machineoperator elke keer een minuut moet wachten voordat het paneel gefreesd kan worden. We gaan testen bij welke input het algoritme snel genoeg is en wanneer dit niet het geval is.

Bij de tests gaan we kijken naar rechthoekige panelen die onderling verschillen op het gebied van:

- Afmeting van het paneel
- Het aantal verboden punten
- Het aantal bewerkingspunten

Door te variëren met bovenstaande eigenschappen is het mogelijk om te zien voor welke panelen het algoritme veel tijd nodig heeft. Hierbij zijn de verboden en bewerkingspunten willekeurig aan het paneel toegewezen. We hebben echter al gezien dat het van belang kan zijn wat de precieze samenstelling van deze punten is. Waar in figuur 5.3 slechts 3 seconde nodig was om een patroon te berekenen, was dit met een andere samenstelling van deze punten ruim 10 maal zo groot in figuur 5.11. We willen uitsluiten dat we toevallig met een makkelijke of moeilijke samenstelling van verboden en bewerkingspunten te maken hebben. Dit lossen we op door het algoritme meerdere malen naar hetzelfde paneel te laten kijken, alleen dan met elke keer een nieuwe verzameling willekeurige verboden en bewerkingspunten. Van al deze keren houden we de looptijd van het algoritme bij, waarmee we een gemiddelde looptijd kunnen bepalen.

Als eerste kijken we of het aantal verboden en bewerkingspunten van grote invloed zijn op de looptijd van het algoritme. Hiervoor nemen we meerdere panelen met een grootte van 50 bij 50 cm waarop verschillende aantallen verboden en bewerkingspunten aanwezig zijn. Voor al deze panelen laten we het algoritme optimale cuppatronen berekenen. Uit de gemiddelde looptijd van algoritme kunnen we dan afleiden hoe groot de invloed van deze punten is. De resultaten hiervan zijn te zien in tabel 5.1

Tabel 5.1

Afmeting (breedte bij lengte cm)	Aantal verboden punten	Aantal bewerkings punten	Aantal runs	Gemiddelde tijd (sec)
50 bij 50	0	0	50	0.06
50 bij 50	3	3	50	0.07
50 bij 50	5	5	50	0.06
50 bij 50	10	0	50	0.06
50 bij 50	0	10	50	0.07

Tabel 5.1 laat zien dat bij een relatief klein paneel het aantal verboden en bewerkingspunten weinig invloed heeft op de snelheid van het algoritme. Dit gaan we ook testen voor een paneel met een afmeting van 100 bij 100 cm. Tabel 5.2 laat hier de uitkomsten van zien.

Tabel 5.2

Afmeting (breedte bij lengte cm)	Aantal verboden punten	Aantal bewerkings punten	Aantal runs	Gemiddelde tijd (sec)
100 bij 100	0	0	50	0.27
100 bij 100	5	5	50	0.64
100 bij 100	10	10	50	0.60
100 bij 100	20	0	50	1.03
100 bij 100	0	20	50	0.91

Bij een paneel van 100 bij 100 cm is zichtbaar dat het algoritme er daadwerkelijk langer over doet als er verboden of bewerkingspunten op het paneel aanwezig zijn. Echter valt de gemiddelde looptijd nog ruim binnen de toegestane tijd. Bij alle 250 runs was er zelfs geen uitschieter bij die daar in de buurt kwam (geen enkele looptijd zat boven de 2 seconden).

De uitkomsten uit de tabellen 5.1 en 5.2 lijken tegenstrijdig met wat we gezien hebben in figuur 5.11. Daar was een verandering van de verboden en bewerkingspunten immers de oorzaak van een 10 keer zo grote looptijd. Voor deze flinke toename kan echter nog andere een oorzaak voor zijn. Als we de figuren 5.3 en 5.11 goed bekijken zien we dat niet alleen de verzameling van verboden en bewerkingspunten en de posities van de cups zijn veranderd. Het aantal balken dat het paneel stabiel houdt is ook toegenomen van zes balken in figuur 5.3 naar zeven balken in figuur 5.11.

We gaan testen of dit echt zoveel invloed heeft op de looptijd. Dit doen we door te kijken naar panelen van dezelfde breedte (100 cm) maar met verschillende lengtes. Voor langere panelen zouden ook meer balken nodig moeten zijn om ze stabiel te houden. Om uit te sluiten dat de verboden of bewerkingspunten invloed hebben, zijn deze niet aanwezig op de panelen. De uitkomsten zijn te zien in tabel 5.3. Hierin staat ook aangegeven hoeveel balken er in de oplossing gebruikt worden.

Tabel 5.3

Afmeting (breedte bij lengte cm)	Aantal balken	Aantal runs	Gemiddelde tijd (sec)
100 bij 100	4	20	0.27
100 bij 110	5	20	1.77
100 bij 120	5	20	3.97
100 bij 130	5	20	7.75
100 bij 140	6	20	8.52
100 bij 150	6	20	1.71
100 bij 160	6	20	4.60
100 bij 170	7	20	24.16
100 bij 180	7	20	2.01
100 bij 190	7	20	1.01
100 bij 200	8	20	133.13

Tabel 5.3 laat gedeeltelijk hetzelfde effect zien als wat we uit figuren 5.3 en 5.11 hebben geconcludeerd. Vooral de panelen met een lengte van 170 en 200 cm zorgen voor een grote looptijd van het algoritme. Bij deze panelen is er, ten opzichte van een 10 cm korter paneel, één extra balk nodig om het paneel stabiel te houden.

Om duidelijk te maken wat volgens ons de oorzaak van deze lange looptijd is vergelijken we de panelen met een lengte van 190 en 200 cm met elkaar. Bij een lengte van 190 cm heeft het algoritme vrij snel een oplossing gevonden die uit zeven balken bestaat. Daarnaast is het snel duidelijk dat een oplossing met zes balken niet tot de mogelijkheden behoort, waarna het algoritme wordt beëindigd. Bij het paneel met een lengte van 200 cm komt het algoritme (waarschijnlijk) wederom binnen korte tijd met een toelaatbare oplossing van het probleem. Deze oplossing bestaat echter uit acht balken. Het kost daarna erg veel tijd om erachter te komen dat een oplossing met zeven balken niet mogelijk is.

In paragraaf **4.2** hebben we het over een approximatie methode gehad die we zouden gaan toepassen wanneer bleek dat de looptijd van het algoritme te groot zou zijn. Bij deze methode kunnen we de optimale oplossing tot een factor $(1 + \varepsilon)$ benaderen. In het geval dat we net besproken hebben heeft dit echter geen zin. Dit paneel wordt al stabiel gehouden door acht balken en meer hebben we er niet tot onze beschikking. We zouden op een niet toegestaan patroon uitkomen wanneer we de oplossing gaan benaderen.

Daarnaast doet het probleem met een lange looptijd zich volgens de tests alleen voor bij het gebruik van bijna alle balken van het freesbed. We kunnen ons afvragen wat dan het nut van het minimaliseren van het aantal balken is, aangezien er toch geen ruimte is om meerdere panelen tegelijk op het freesbed te leggen. We zouden in deze situatie ook genoegen kunnen nemen met een toelaatbare oplossing in plaats van een optimale.

Deze toelaatbare oplossingen kunnen we vinden door slechts een kleine aanpassing te maken aan ons huidige model. Het ILP model uit paragraaf **4.3** heeft de volgende doelfunctie:

$$\min \sum_{j=1}^{m} c_j y_j,$$

waarbij m gelijk is aan het aantal kolommen van de discretisatie van het gehele freesbed. In ons geval geldt $m = 28$. De kostenfunctie c ziet er als volgt uit:

$$c_j = 1, \qquad \forall j \leq m_1$$

$$c_j = 0, \qquad \forall j > m_1$$

waarbij m_1 gelijk is aan het aantal kolommen van het paneel. In de hierboven besproken gevallen is m_1 gelijk aan 17 en 20. Deze doelfunctie gaan we aanpassen zodat het ILP alleen nog maar een toelaatbare oplossing hoeft te vinden. Dit doen we door de kostenfunctie op de volgende manier aan te passen:

$$c_j = 1, \qquad \forall j$$

De kosten voor het plaatsen van een balk is nu voor alle kolommen gelijk. Aangezien het aantal balken dat deel uitmaakt van het freesbed constant is, zal de waarde van de doelfunctie altijd gelijk zijn aan acht. Het algoritme hoeft er nu alleen nog voor te zorgen dat de oplossing voldoet aan de restricties.

Volgens onze argumentatie zou het algoritme wel snel genoeg een toelaatbare oplossing kunnen vinden, maar kost het veel tijd om te kijken of de oplossing optimaal is. Dit gaan we testen door voor de panelen met een grote looptijd uit tabel 5.3 een toelaatbare oplossing te bepalen in plaats van een optimale. De uitkomsten hiervan zijn terug te vinden in tabel 5.4

Tabel 5.4

Afmeting (breedte bij lengte cm)	Aantal runs	Gemiddelde tijd (sec)
100 bij 170	20	0.19
100 bij 200	20	1.55

Tabel 5.4 laat zien dat met deze kleine aanpassing van het model, de looptijd van het algoritme drastisch verlaagd wordt. Hierdoor kunnen we ook voor panelen die stabiel gehouden moeten worden met veel balken snel een patroon van cups berekenen.

Conclusie

In dit hoofdstuk hebben we voor verschillende panelen een patroon van cups gecreëerd. Hierbij hebben we gekeken naar de tijd die het algoritme erover deed om tot de oplossing te komen. Bij panelen die stabiel gehouden moeten worden met relatief veel balken kan het voorkomen dat de looptijd van het algoritme te groot wordt. In dit geval kunnen we de kostenfunctie van het model aanpassen zodat er naar een toelaatbare oplossing gezocht wordt in plaats van naar een optimale. Dat dit een patroon oplevert waarin niet een minimaal aantal balken gebruikt wordt is geen probleem, aangezien er in deze situaties toch weinig ruimte is om op het freesbed een tweede paneel vast te houden. Door te zoeken naar alleen een toelaatbare oplossing kunnen we ook voor deze panelen snel een patroon van cups bepalen.

6. Conclusie

Het doel van dit onderzoek is om een algoritme te ontwikkelen dat een patroon van cups kan creëren voor alle panelen die Kast op Maat moet bewerken. De cups moeten ervoor zorgen dat het paneel stabiel blijft tijdens het frezen. Daarnaast is het van belang dat het algoritme redelijk snel een patroon voor een paneel kan berekenen. Als extra eis stellen we dat het algoritme een patroon moet vinden dat zo min mogelijk balken gebruikt. Door te zoeken naar deze optimale patronen kunnen we eventueel meerdere panelen tegelijkertijd op het freesbed leggen.

Het continue plaatsingsprobleem hebben we gediscretiseerd door elk paneel te modelleren als een verzameling punten. We hebben een stabiliteitsvoorwaarde geformuleerd om ervoor te zorgen dat de gecreëerde patronen het paneel stabiel houden tijdens het frezen. In de loop van het onderzoek hebben we deze voorwaarde uitgebreid door ook rekening te houden met plekken op het paneel waar een bewerking uitgevoerd moet worden.

We zijn begonnen met het kijken naar een paneel dat dusdanig smal is dat het slechts met één cup per balk vastgehouden kan worden. Met behulp van dynamisch en lineair programmeren kunnen we voor deze panelen optimale patronen vinden. We hebben aangetoond dat dit patroon efficiënt berekend kan worden door te bewijzen dat de restrictiematrix van het bijbehorende LP totaal unimodulair is.

Vervolgens hebben we gekeken naar panelen die stabiel gehouden moeten worden door twee cups per balk, waarbij we een gesimplificeerde stabiliteitsvoorwaarde gebruikt hebben. Voor deze panelen kunnen we optimale patronen berekenen door middel van dynamisch en lineair programmeren. Door een stelling van Hoffman en Schwartz toe te passen hebben we aangetoond dat het LP horende bij deze panelen altijd een geheeltallige oplossing levert.

Met de originele stabiliteitsvoorwaarde is dit voor deze, en grotere, panelen niet het geval, waardoor we een geheeltallige oplossing moeten forceren. Bij het bepalen van de patronen kunnen we dan geen gebruik meer maken van een efficiënte LP solver. In plaats daarvan maakt het algoritme dat we ontwikkeld hebben gebruik van de optimalisatiesoftware CPLEX, die ILPs op kan lossen.

Door het algoritme te testen voor verschillende panelen hebben we laten zien onder welke omstandigheden de looptijd ervan klein genoeg is. Wanneer het algoritme te lang over een oplossing moet nadenken, kunnen we er met een kleine aanpassing voor zorgen dat er gezocht wordt naar slechts een toelaatbare oplossing in plaats van een optimale. We hebben laten zien dat deze toelaatbare oplossing wel snel gevonden kan worden.

7. Discussie

In dit onderzoek hebben we gekeken naar de efficiëntie van de gebruikte oplosmethode. Dit hebben we gedaan omdat het algoritme niet bruikbaar zou zijn wanneer de machineoperator elke keer lang op een cuppatroon moet wachten. Hierbij gaan we ervan uit dat dit patroon ter plekke berekend moet worden. Een andere optie is om voor alle panelen de patronen centraal te laten berekenen en deze dan via een website door te sturen. Via deze website krijgt de CNC-machine op dit moment al alle eigenschappen van de te bewerken panelen doorgestuurd. Het daarnaast doorgeven van de cuppatronen lijkt slechts een kleine aanpassen te zijn van de huidige situatie. Het voordeel hiervan is dat er dan meer tijd beschikbaar is om de cuppatronen te berekenen. Daarnaast hoeft dan niet op elke bij een CNC-machine horende computer de benodigde software geïnstalleerd te worden die bij het uitvoeren van het algoritme gebruikt wordt.

Om ervoor te zorgen dat het algoritme patronen van cups creëert die de panelen stabiel houden, hebben we een stabiliteitsvoorwaarde geformuleerd. Wanneer een patroon aan deze voorwaarde voldoet dan wordt het paneel stabiel gehouden. In hoofdstuk **3** hebben we deze voorwaarde nog uitgebreid door rekening te houden met bewerkingspunten. Het algoritme geeft ons uiteindelijk een patroon van cups dat aan deze eisen voldoet. Echter zijn er vaak meerdere patronen die aan deze eisen voldoen. We zouden al deze optimale patronen nog kunnen onderscheiden en daarvan een ‘meest gewenste’ kiezen. Een keuze die we kunnen maken is bijvoorbeeld kiezen voor het optimale patroon waar de cups zo ver mogelijk uit elkaar staan. Een patroon waar de cups verder uit elkaar staan is vaak stabieler dan een patroon met cups die zich allemaal dicht bij elkaar bevinden.

We zijn er in dit onderzoek van uitgegaan dat er alleen hele cups gebruikt worden in het cuppatroon. Het is interessant om te kijken wat er zou veranderen aan het model als ook halve cups toegelaten worden. Deze zouden bijvoorbeeld gebruikt kunnen worden op een positie waar geen plek is om een hele cup te plaatsen.

8. Referenties

[1] www.kastopmaat.nl ingezien op 14-09-2011

[2] B. Korte , J. Vygen; Combinatorial optimization theory and algorithms 2nd ed, Springer, 2002

[3] A. J. Hoffman, D. E. Schwartz; On lattice polyhedra, In Combinatorics (Proc. Fifth Hungarian Colloq) ,1976

[4] S. Masuyama, T. Ibaraki, T. Hasegawa; The computational complexity of the M-center problems in the plane, Trans. IECE Japan E64 57-64, 1981

[5] T. Nieberg, J. Hurink; A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs, Memorandum No. 1732, 2004

Bijlage A

Dit is de uitwerking van het dynamisch programmeren horende bij het voorbeeld uit sectie **2.3**. Het probleem heeft de volgende eigenschappen:

- Punten $j = \{1,2, \dots ,18\}$
- Maximale afstand $max = 2$
- Verzameling van verboden punten $j = \{3,4,5,7,10,11,13,17\}$

We beginnen aan de rechterkant ($j = 18$) van het paneel. De waarden in de tabel zijn de uitkomsten van het minimaliseren over de beslissingen. Bij sommige nummers staat een asterisk. Dit betekent dat het plaatsen van een cup in deze situatie verplicht is. In de eerste kolom ($j = j$) is de standaard recursieve formule te zien die geldt voor alle j, uitgezonderd voor $j = 18$ en $j = 1$. De formules hiervoor zijn terug te vinden in sectie **2.2**.

j	$j = j$	$j = 18$	$j = 17$	$j = 16$
$k = 1$	$V_j(1) = \min\{1 + V_{j+1}(1), 0 + V_{j+1}(2)\}$	0	0	1
$k = 2$	$V_j(2) = \min\{1 + V_{j+1}(1), 0 + V_{j+1}(3)\}$	0	1	1*
$k = 3$	$V_j(3) = \min\{1 + V_{j+1}(1), 0 + V_{j+1}(4)\}$	1*	-	1*
$k = 4$	$V_j(4) = \min\{1 + V_{j+1}(1), 0 + V_{j+1}(5)\}$	-	-	1*
$k = 5$	$V_j(5) = 1 + V_{j+1}(1)$	-	-	1*

j	$j = 15$	$j = 14$	$j = 13$	$j = 12$	$j = 11$
$k = 1$	1	1	1	1	2
$k = 2$	1	1	1	2	2
$k = 3$	1	1	2	2	2
$k = 4$	1	2	2	2*	2
$k = 5$	2*	2*	-	2*	-

j	$j = 10$	$j = 9$	$j = 8$	$j = 7$	$j = 6$
$k = 1$	2	2	2	3	3
$k = 2$	2	2	3	3	3
$k = 3$	2	3*	3	3	3
$k = 4$	-	3*	3	3	4*
$k = 5$	-	3*	3*	-	4*

j	$j = 5$	$j = 4$	$j = 3$	$j = 2$	$j = 1$
$k = 1$	3	3	4	4	-
$k = 2$	3	4	4	5*	-
$k = 3$	4	4	-	5*	5
$k = 4$	4	-	-	5*	-
$k = 5$	-	-	-	5*	-

Zoals we al gezien hadden in **2.3**, bestaat het optimale patroon van cups uit 5 balken. Door in de tabel het pad terug te lopen waar we een asterisk bij hebben gezet, kunnen we zien hoe de optimale oplossing eruit ziet: $j = \{2,6,9,14,16\}$. Soms maakt het voor de optimale oplossing niet uit of er wel of geen cup geplaatst wordt. Er zijn dan meerdere optimale paden door het tabel heen. Onze oplossing is gelijk aan het tweede paneel in figuur 2.4.

Bijlage B

In paragraaf **2.3** is een voorbeeld van een te bewerken paneel gegeven. Bijlage **A** laat zien hoe we met dynamisch programmeren een cuppatroon voor dit paneel kunnen berekenen. Hier gaan we hetzelfde doen door het bijbehorende LP op te lossen.

$$\min \sum_j x_j \qquad \text{(minimaliseer het aantal balken)}$$

$$s.t. \sum_j a_{ij}\, x_j \geq 1, \qquad \forall i \qquad \text{(alle punten zijn gecoverd, stabiliteitsconditie)}$$

$$0 \leq x_j \leq 1, \qquad \forall j \qquad \text{(elk punt krijgt wel of geen cup)}$$

Matrix $A = [a_{ij}]$ zorgt ervoor dat elk punt wordt gecoverd. Voor dit probleem zien deze restricties er als volgt uit:

$$\begin{bmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1
\end{bmatrix}
\begin{bmatrix} x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8\\x_9\\x_{10}\\x_{11}\\x_{12}\\x_{13}\\x_{14}\\x_{15}\\x_{16}\\x_{17}\\x_{18}\end{bmatrix}
\geq
\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix}$$

De verboden punten van het paneel zijn terug te zien als de nulkolommen in de matrix A. De maximale afstand in dit voorbeeld is gelijk aan twee. Dat betekent dat punt j gecoverd kan worden door vijf verschillende punten; namelijk punten $\{j-2,\ j-1,\ j,\ j+1,\ j+2\}$. Dit is terug te zien aan de vijf enen in elke kolom. Uitzondering hierop zijn de punten aan de rand van het paneel.

Dit LP kunnen we efficiënt oplossen met een LP solver. Zoals we al eerder gezien hebben, zijn voor dit probleem meerdere oplossingen mogelijk. De solver die wij gebruiken geeft

$$\sum x_j = 5, \qquad x_2 = x_6 = x_9 = x_{12} = x_{16} = 1$$

als oplossing. Deze is gelijk aan het eerste paneel in figuur 2.4.