Binnen het thema ‘wiskunde in de fabriek’ hield **Jan Verbakel**, naar eigen zeggen wiskundige maar vooral puzzelaar, een voordracht op de Nationale Wiskunde Dagen 2011 over de wiskundige hersenbrekers waarmee men wordt geconfronteerd bij de productie van plaatsingsmachines. Varianten van het Travelling Salesman Problem passeren de revue, waarbij het resultaat verrast en het plezier ervan afspat.

Een doos vol puzzels

Optimalisering van een plaatsingsmachine

Inleiding

Een groot gedeelte van mijn werkzame leven heb ik bij Philips doorgebracht, de eerste twintig jaar op het NatLab, en daarna bij allerlei meer productgerichte afdelingen. Het laatst heb ik bij Assembléon gewerkt aan plaatsingsmachines. Ik ben sinds drie jaar met pensioen, maar af en toe ben ik nog een beetje aan het werk.

Dit artikel gaat in de eerste plaats over puzzels, maar ook over wiskunde. Ik heb het afgelopen jaar als wiskundige gewerkt bij DIMA, een fabrikant van plaatsingsmachines. Het gaat dus ook een beetje over plaatsingsmachines. In de loop van dit verhaal zal u hopelijk duidelijk worden wat zij doen en waarom ze interessant zijn voor wiskundigen. Kort samengevat: plaatsingsmachines plaatsen elektronische componenten op bordjes (ook wel printplaten genoemd). De software in de plaatsingsmachine moet ervoor zorgen dat dit plaatsen zo snel en nauwkeurig mogelijk gebeurt. Het vakgebied dat zich (binnen de wiskunde) bezighoudt met dit soort problemen heet optimalisatie.

De doos met puzzels, die uit optimalisatie van plaatsingsmachines voortkomt, bevat vooral problemen op het gebied van modellering (wiskunde, het snappen) en op het gebied van algoritmes (informatica, het oplossen). Een waarschuwing vooraf: verwacht van mij geen zware wiskunde, ik bedrijf de wiskunde als puzzelaar.

Travelling Salesman Problem

Het bekendste probleem in de optimalisatie is het TSP, het Travelling Salesman Problem of (in nog beter Nederlands) het probleem van de handelsreiziger. Een aantal steden moet worden bezocht door een handelsreiziger. Het probleem is: vind de kortste route. Meestal zijn het begin en einde van de route gegeven, soms zijn begin en einde dezelfde plaats.

TSP : mijn biebs in Belgisch Limburg

Een eerste voorbeeld. Op deze kaart van Belgisch Limburg zijn mijn favoriete bibliotheken aangegeven (met zwarte stippen, red.). Beginnend en eindigend in Lommel zijn er 10! = meer dan 3 miljoen mogelijke routes. Dit probleem gaan we vandaag niet oplossen.

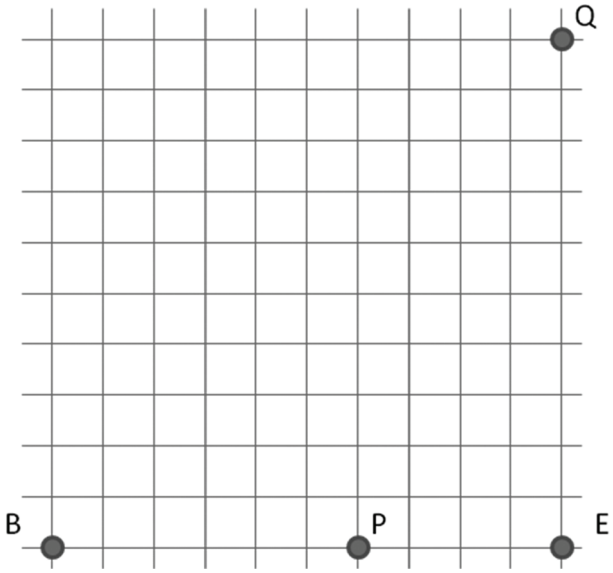


fig. 1

In figuur 1 zien we vier punten: het beginpunt B, de tussenpunten P en Q en het eindpunt E. De vraag is nu: “Wat is de kortste route van B naar E via P en Q?”Natuurlijk is de route $BPQE$ de kortste, dat zie je

met het blote oog. Tenminste dat denk ik. ***BPQE*** levert 26.77; ***BQPE*** levert 28.91 en we zien dat ***BPQE*** dus inderdaad minder is.

We zullen dadelijk nog een keer op dit overdreven simpele probleem terugkomen. Maar eerst een kort overzicht van de inhoud. Allereerst kijken we naar plaatsingsmachines: de hardware, de onderdelen en vooral hun manier van werken. Dat zal ons een eerste kennismaking opleveren met een bron van puzzels, die diep vanuit de wiskunde blijkt te komen. Daarna houden we ons bezig met de software van plaatsingsmachines, vooral optimalisatie. En daarbij barst het feest pas echt los: uit alle uithoeken van het optimalisatieproces komen puzzels tevoorschijn. De puzzels gaan over de diverse onderdelen van de plaatsingsmachine. Van sommige bekijken we alleen een samenvatting, andere werken we verder uit. Door het gehele artikel heen komt u TSP-puzzels tegen. Wiskunde vindt u overal terug, hoop ik. Veel plezier.

Plaatsingsmachines

Allereerst een mondelinge demonstratie. Hopelijk verduidelijkt die al een beetje de werking van zo'n plaatsingsmachine. Componenten of chips worden aangeleverd in een *feeder* (voederbak). De chips worden opgezogen en vastgehouden door een *tool* dat onder een pipet hangt, getransporteerd door de kop en dan geplaatst op een bord. De meeste plaatsingsmachines werken met zuigkracht. Ik zal proberen dat aan de hand van onderstaand voorbeeld duidelijk te maken.

Rolverdeling	
Feeder	Bakje
Chip	Chip
Pipet	Mond
Tool	Rietje
Kop	Hoofd
Bord	Bord

De rolverdeling kunt u hierboven lezen. De feeder wordt gespeeld door dit bakje, de chip door de (Pringles) chip, de pipet door mijn mond, de tool door een rietje, de kop door mijn hoofd, en het bord door het bord:

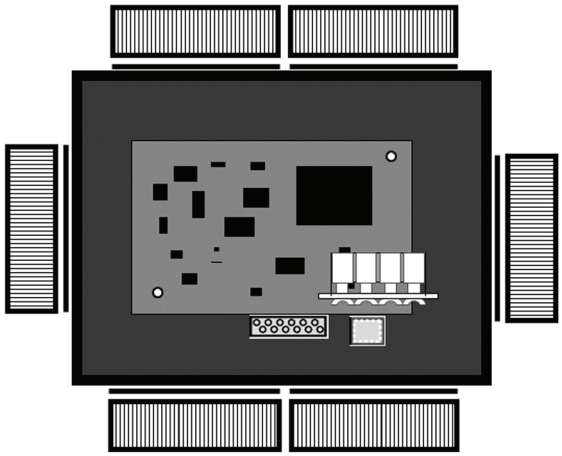
> Ik pak het rietje in mijn mond, ga met mijn hoofd naar het chipsbakje, zuig met mijn mond een chip vast aan het rietje, plaats mijn hoofd boven het bord, leg het chipje op het bord en stop met zuigen.

Het gebeurt op die manier wat langzaam en wat onnauwkeurig. Daarom laten we dit soort werk aan een machine over: die kan dat veel beter, veel sneller (6000 componenten per uur) en veel nauwkeuriger (50 mu= 0,05 mm). En ons werk wordt dan intelligenter: we programmeren de machine zodanig dat hij nog sneller en nauwkeuriger wordt; zoals we zojuist al zeiden, optimalisatie.

DIMA in Deurne is het bedrijf waar ik het afgelopen jaar heb gewerkt. DIMA verkoopt een lijn van machines die bij het plaatsingsproces betrokken zijn. In totaal werken er zo'n vijfentwintig mensen, waarvan een groot gedeelte bij ontwikkeling. DIMA is op internet te vinden op http://www.dimagrp.com. Hier op papier zullen we de plaatsingsmachine beschrijven aan de hand van een plaatje.

Plaatsingsmachine: plattegrond

Hier zien we een plattegrond van de MP200-machine. De donkere rechthoek is de binnenkant van de machine.



Rondom de machine hangt een aantal *feeders* (bakjes). Iedere feeder bevat één componenttype ('smaak chip'), maar meestal wel een groot aantal daarvan.
De feeders hangen aan *feeder banken*, op voorgedefinieerde *feeder posities*. Aan de buitenkant van de machine zien we in het plaatje de 6-feeder. De componenten moeten geplaatst worden op het bord, de lichtgrijze rechthoek. Het witte gevaarte boven het bord is de kop met vier pipetten en vier tools.

De tools (rietjes) pakken de componenten uit de feeders. De kop transporteert de vier componenten naar het bord, waar ze door de tools worden geplaatst. Bij de demonstratie diende mijn hoofd als kop, mijn mond als pipet, en een rietje als tool. Dit hoofd heeft dus vier monden en kan dus ook vier rietjes hebben. Ieder tool houdt een component vast, en de kop kan dus vier componenten vervoeren.

Je kunt je voorstellen dat je voor het vervoer van grote chips een breder rietje gebruikt, en voor de kruimeltjes

een dun rietje. Er zijn zo'n tien verschillende tool types (soorten rietjes). Tools kunnen gewisseld worden, er is een magazijn aanwezig, waarin de tools zitten die nu niet gebruikt worden. Van de meest gebruikte tools zijn meerdere exemplaren aanwezig. Wij zien die toolwisselaar getekend net onder het bord.

Samenvattend: de componenten worden stuk voor stuk opgezogen uit de feeders door de tools. Dan reist de kop naar het bord, en dan worden de componenten stuk voor stuk op het bord neergelegd. Die bewegende kop zorgt voor de XY-beweging in de machine. Hij werkt met twee (even snelle) motoren: een voor de X-richting en een voor de Y-richting.

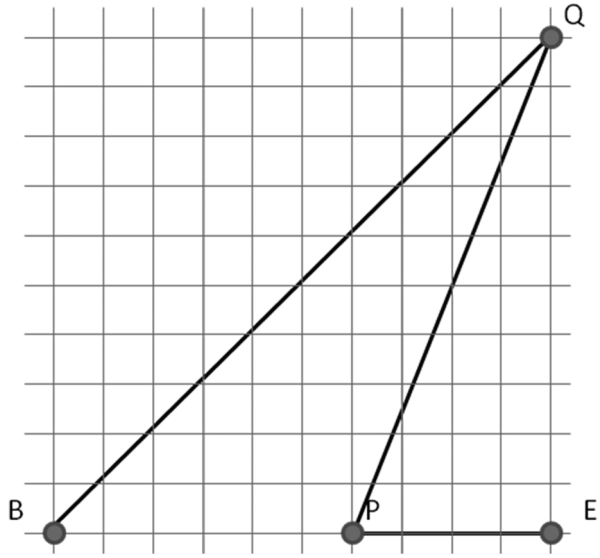
Puzzelalarm

De opmerking over die XY-beweging met twee motoren doet een belletje rinkelen. Het feit dat deze twee motoren onafhankelijk zijn, heeft grote gevolgen voor berekeningen van afstanden. Het betekent dat de afstand van twee punten gemeten moet worden als het maximum van de X-afstand en de Y-afstand: $\max(dx, dy)$. Maar dat is de supremum-norm.

Twee vragen vechten nu om voorrang:
1. Hoe zien de punten van gelijke afstand eruit?
2. Wat gebeurt er met de TSP van vier punten?

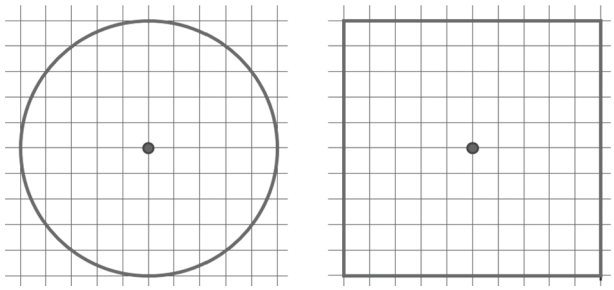
De tweede vraag heeft gewonnen : Wat gebeurt er met de TSP van vier punten? Als we de afstanden volgens de supremum-norm uitrekenen, krijgen we:

$BPQE$ levert $6 + 10 + 10 = 26$;
$BQPE$ levert $10 + 10 + 4 = 24$;
en we zien dat $BQPE$ nu wint.



We zien dat de supremum-norm een andere winnaar heeft! Dat betekent niet alleen dat we in onze software een ander algoritme moeten gebruiken om de afstand uit te rekenen, maar ook dat we onze intuïtieve gedachten niet meer zo goed kunnen vertrouwen.

De andere vraag was : Hoe zien de punten van gelijke afstand eruit ? Hier hebben we ze getekend. We zien dat de Euclides-norm en de supremum-norm er echt anders uitzien. De cirkel $dx^2 + dy^2 = 1$ van Euclides versus het vierkant $\max(dx, dy) = 1$ van de supremum. Straks, bij een van de puzzels, zullen we dit nog gebruiken.



Sorry voor deze onderbreking van de verhaallijn. Ik moest dat nieuws even kwijt; terug naar ons verhaal... We waren gebleven bij de plattegrond van de machine. We hebben straks de onderdelen van de hardware bekeken.

De feeders aan de buitenkant, de bewegende kop, de pipetten, de tools, het bord en de componenten die uit de feeders worden gepakt door de pipet met tool, worden getransporteerd naar het bord door de kop, en tenslotte op het bord worden geplaatst.

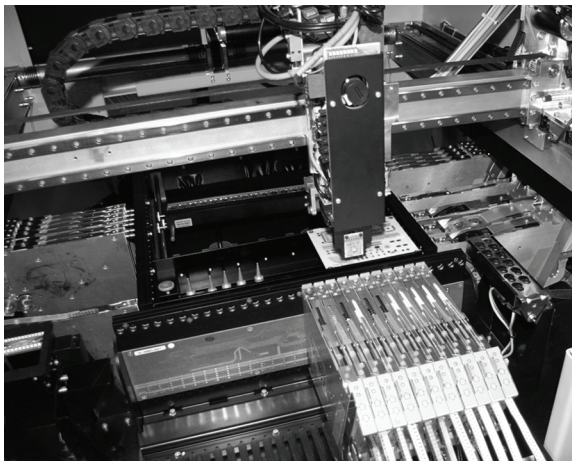
Software

Nu kunnen we kijken naar de software van de plaatsingsmachine. Eerst even op een rijtje zetten wat er gebeurt in de machine. Voor de suggestie hebben we dit al geschreven in een stijl die aan programma's doet denken.

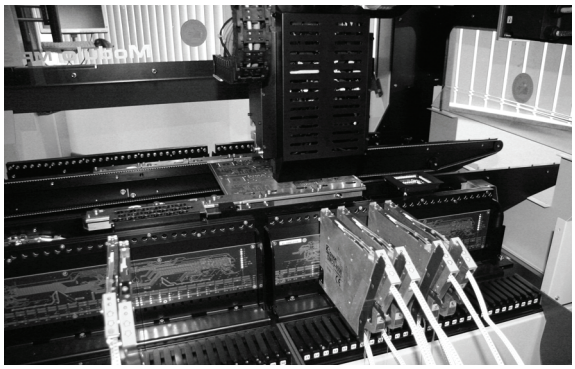
```
Bord komt in de machine
  Herhaal tot het bord vol is
    4 maal
      De kop gaat naar een feeder
      Een tool pakt een component uit de feeder
    4 maal
      De kop gaat naar een positie op het bord
      Een tool plaatst een component op het bord
  Einde herhaling
Bord gaat uit de machine
```

Er komt een bord in de machine. We herhalen, tot het bord vol is, de volgende acties: We gaan met de kop naar een feeder toe, en pakken met een tool een component daaruit. Daarna naar een andere feeder, pakken weer een component. Weer verplaatst de kop zich naar een derde feeder, en weer pakt een tool een component. En tenslotte naar de vierde feeder, en weer een component.

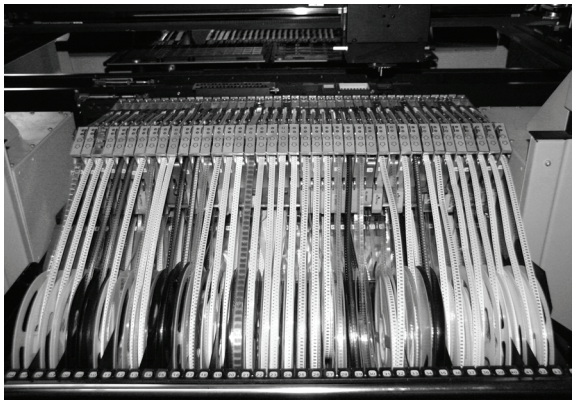
De machine in beeld



De printplaat met daarboven de kop met de pipet, links daarvan een aantal pipetten. De feeders zijn in deze situatie geen bakjes, maar rollen waar de componenten op zitten.



De MP200 in actie.



Tapefeeders van de MP200.

Nu heeft iedere tool een component en gaan we naar het bord toe voor het plaatsingsproces. De kop gaat naar een componentpositie, en een tool plaatst een component. Daarna naar een volgende componentpositie, en weer plaatst een tool een component. U begrijpt het al: ook naar een derde en vierde positie, en ook daar worden componenten geplaatst.

Nu zijn alle tools weer leeg, en kan een nieuwe run van vier componenten beginnen. Weer eerst vier componenten oppakken en daarna vier componenten plaatsen. Tot na verloop van tijd alle componenten geplaatst zijn en het bord naar buiten kan worden gebracht en het volgende bord binnengehaald wordt.

Aansturing van de machine

De aansturing van de machine gebeurt vanuit een PC met Windows, met behulp van speciale software, die alles weet van de machine. De optimalisatie is een onderdeel van die software en loopt op de PC. Op ieder moment kent de optimalisatie de toestand van de machine en kan zich daaraan aanpassen. Dit is vooral bedoeld voor de flexibiliteit, een typisch kenmerk van de DIMA-machines. Een ander voorbeeld van die flexibiliteit is het volgende. Er wordt per run (vier componenten oppakken en vier componenten plaatsen) geoptimaliseerd. Dus terwijl er vier componenten gepakt en geplaatst worden, wordt uitgerekend welke vier componenten hierna aan de beurt zijn. Dat optimaliseren per run heeft ook zijn schaduwzijde. We hebben ongeveer een halve seconde voor dit karwei, iedere run. En dat is vrij kort.

Optimalisatie

En daarmee zitten we midden in de optimalisatie. Optimalisatie kan in die beperkte tijd niet 'gewoon' alle mogelijke runs uitrekenen en kijken welke de snelste is. Het gaat dan over vreselijke getallen, vooral omdat we vier pipetten hebben. Alle aantallen worden onmiddellijk tot de vierde macht verheven. De snelheid werd op twee manieren verbeterd. Allereerst werd het optimalisatieproces gesplitst in een aantal opeenvolgende keuzes. (Een gedeelte van die splitsing was al aanwezig, hij werd uitgebreid en bewuster gekozen.) Per run gebeurt het volgende:

- Kies tools
- Kies feeders (inclusief componenttypes)
- Kies plaatsingen
- Bepaal pick-uproute
- Bepaal plaatsingsroute

De andere verbetering was het platslaan van ieder van de stappen. Bij de stappen waren de aantallen die uitgerekend werden nog vaak te groot, vooral omdat het vierde machten waren. Als we vier feeders bepalen, en we proberen alle mogelijkheden, dan krijgen we N^4 mogelijkheden, met N het aantal feeders en dat kan meer dan 100 zijn. Mijn voornaamste opdracht, toen ik werd ingehuurd bij DIMA was: dat aantal moet omlaag. Daartoe hebben we vooral dat platslaan beoefend: maak de algoritmes $O(N^2)$ in plaats van $O(N^4)$. Maar er zijn ook nog enkele nieuwe problemen uitgekomen, ook veroorzaakt doordat we nu vier pipetten hebben.

De puzzels

De puzzels zullen we bespreken per stap uit de keuzes.

Puzzel van tools: urgentie

Voor de toolkeuze moeten we vooral vooruitkijken, naar urgentie. Een voorbeeld:

	T1	T2	#C
#Tools	4	2	
C1	+	-	4
C2	-	+	4

We hebben een bordje met acht componenten. We hebben twee componenttypes, C1 en C2. Van beide types willen we vier componenten plaatsen. De machine heeft vier pipetten. We hebben vier tools van type T1 en twee tools van type T2. Componenttype C1 wordt geplaatst met tooltype T1. Componenttype C2 wordt geplaatst met tooltype T2. We beginnen lompweg. We zetten eerst vier tools T1 op de pipetten, die de vier componenten C1 gaan plaatsen. Daarna hebben we nog vier componenten C2 te plaatsen, maar maar twee tools T2. We zetten nu deze twee tools op de pipetten en plaatsen twee componenten C2. Meer componenten kunnen we deze run niet plaatsen. De run erna plaatsen we ook weer twee componenten C2, en we zijn klaar. Maar we hebben nu wel drie runs gedraaid. Kan dat niet korter?

De oplossing wordt duidelijker als we de tabel uitbreiden.

	T1	T2	#C	#T	#R
#Tools	4	2			
C1	+	-	4	4	1
C2	-	+	4	2	2
totaal	+	+	8	4	2

We voegen twee kolommen toe : #T en #R.
#R staat voor aantal runs. Dit is eenvoudig te berekenen uit #R(CT) = #C(CT)/#T(CT), waarin #T staat voor het aantal beschikbare tools voor dit componenttype. We zien:

#R(C1) = 4/4 = 1,
#R(C2) = 4/2 = 2,
#R(C1+C2) = 8/4 = 2.

Dit zegt dus dat we twee runs nodig hebben. In totaal (C1+C2) hebben we die twee runs nodig, en voor C2 hebben we ook twee runs nodig. We moeten dus direct aan C2 beginnen! We zetten twee tools T2 op de pipetten en vullen de andere twee met twee tools T1. We draaien nu twee runs, met in beide runs twee componenten C1 en twee componenten C2. Nu hebben we maar twee runs gebruikt, en niet tussentijds van tools gewisseld. En dat alles kunnen we gemakkelijk aflezen uit de laatste kolom van de tabel. Die tabel werkt trouwens ook als we veel componenttypes hebben en veel tooltypes. De puzzel van urgentie heeft nog een flink aantal complicaties:

– Componenttypes kunnen geplaatst worden door meerdere tooltypes.
– Tooltypes kunnen meerdere componenttypes plaatsen.
– Er kan zelfs overlap zitten tussen de tooltypes van twee componenttypes.

Het idee van deze puzzel was vooral vooruitzien: we willen straks geen vertragingen oplopen. De puzzel van de feederkeuze zullen we straks bespreken.

Puzzel van plaatsingen

We gaan nu de componentplaatsingen kiezen. Tevoren hebben we de feeders gekozen, en daarmee ook beslist welke componenttypes we gaan plaatsen. Maar van de meeste componenttypes moet een aantal componenten op het bord geplaatst worden. Nu gaan we dus uitzoeken welke plaatsingen we gaan doen. Het criterium daarbij is vooral de kleine verplaatsing tussen het plaatsen in.

Ook hier hebben we teveel mogelijkheden. Als we alle mogelijke plaatsen gaan proberen, krijgen we een algoritme van orde (N^4). We willen dat verminderen, vooral omdat er bordjes zijn met $N > 100$. We willen en kunnen hier naar orde (N^2) met behulp van een zogenaamd *Greedy* algoritme. Dat bekijkt niet alle mogelijkheden, maar zorgt wel voor een redelijke oplossing in een redelijke tijd. Meer specifiek: we proberen alle plaatsingen van de eerste feeder, en daarbij zoeken we de beste plaatsing van de tweede feeder, de derde feeder en de vierde feeder, maar onafhankelijk van elkaar. Orde wordt nu dus $N \times (N + N + N) = 3N^2$ in plaats van N^4.

De P&P routes: 2 TSP's

Het bepalen van de pick and place-routes kunnen we splitsen in twee TSP's van vier posities: de vier feederposities en daarna de vier plaatsingsposities. Omdat ieder daarvan maar vierentwintig mogelijkheden heeft, kunnen we hierbij gewoon alle routes berekenen en daaruit de beste kiezen.

De puzzel van de feederkeuze

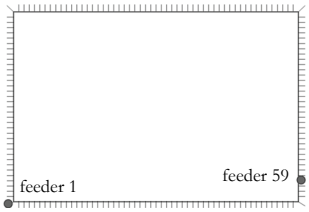
De situatie in het optimalisatieprogramma is als volgt: de tools zijn gekozen en we gaan aan de slag met het kiezen van feeders. Daarbij komen we een aantal problemen tegen. Als we bij vier tools de bijbehorende vier feeders willen bepalen, hebben we in het algemeen (*brute force* dus) een algoritme van orde (N^4).

We hebben het probleem teruggebracht tot een puzzel over feederparen: bereken het beste feederpaar bij

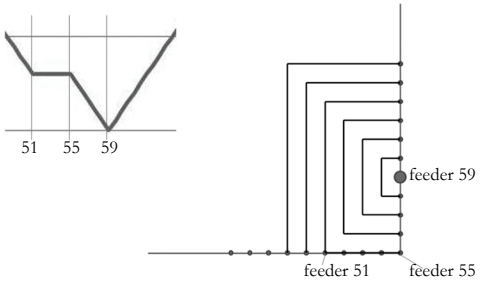
een gegeven toolpaar. Het selecteren gebeurt daarbij op de kleinste afstand van de feeders. We combineren in principe alle feeders van tool $t0$ (op pipet 0) met alle feeders van tool $t1$ (op pipet 1). Het algoritme zou simpelweg kunnen zijn:

– doorloop de lijst van feeders die bij tool 0 mogen,
– doorloop de lijst van feeders die bij tool 1 mogen,
– bereken van alle combinaties de afstand,
– bewaar de combinaties met de kleinste afstanden, daarmee willen we verder.

Dat algoritme is orde (N^2). In sommige gevallen is dat te veel. En het kan duidelijk beter. We hebben, gebruikmakend van de supremum-norm een methode gevonden, die dit proces aanzienlijk kan versnellen. We willen alleen feederparen genereren die dicht bij elkaar liggen. Daartoe zouden we het liefst alle feederposities sorteren. Maar dat geeft niet noodzakelijk een volgorde van afstand van twee feeders. Maar we hebben een eigenschap gevonden die hoop geeft. Daartoe zetten we de feederposities in de volgorde van de rondgang rondom de machine. We lopen daartoe linksom lopend aan de buitenkant van de machine langs de feederposities.

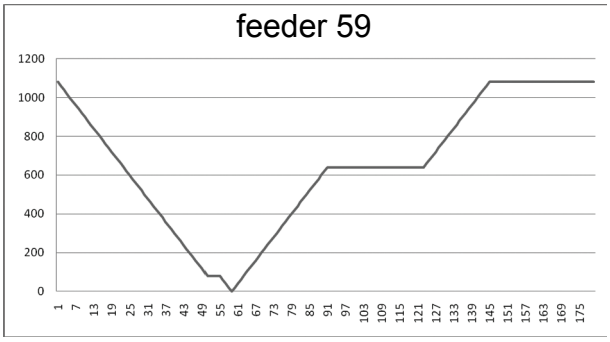


Eerst de feeders aan de voorkant oplopend op X, daarna rechts, oplopend op Y, dan achter, aflopend op X, en tenslotte links aflopend op Y. We eindigen waar we begonnen zijn: linksvoor bij de machine. Daarmee hebben we dus de feeders genummerd van 1 (linksonder) tot en met 180 (weer linksonder). We willen speciale aandacht besteden aan feeder 59, vooral omdat deze vlakbij een hoek ligt.



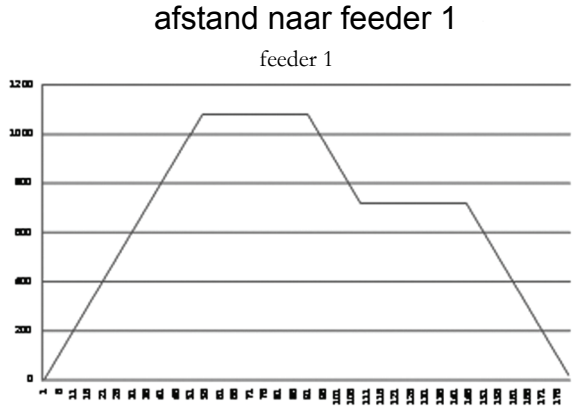
In het rechterplaatje zien we feederpositie 59 nog eens uitvergroot. We zien hoe de afstand vanaf de andere feederposities eruitziet. Aan de bovenkant (60, 61, ..) stijgt de afstand gestaag vanaf 1. Aan de onderkant zien we wat anders. De afstand naar 58 is 1, oplopend tot de afstand naar 55 is 4. Daarna blijft de afstand enige tijd gelijk: alle feeders van 51 tot en met 55 hebben afstand 4 tot feederpositie 59. Dit zien we vooral omdat de feeders 51 tot en met 55 op dezelfde hoogtelijn rondom feeder 59 liggen. Na feeder 51 stijgt de afstand weer: bij 50 is de afstand 5, bij 49 is hij 6 enzovoort.

In het linkerplaatje zien we die afstand ook weergegeven in een grafiek rondom 59. We herkennen de stijgingen en het constante gedeelte.



Op dit plaatje zien we hoe alle afstanden tot feeder 59 eruitzien. We beginnen weer bij feeder 59 zelf, die natuurlijk een afstand 0 heeft. Als we vanuit feeder 59 omhoog lopen, zien we de afstand steeds stijgen (niet dalen) tot aan het gebied tussen feeder 144 en 180. Als we omlaag lopen, zien we de afstand steeds stijgen (niet dalen), tot aan feeder 1, direct naast feeder 180. Wat opvalt is dat de afstand van de feeders naar feeder 59 een soort monotone functie is.

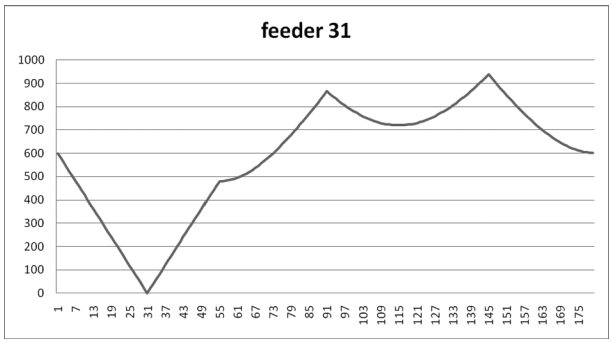
Voor feeder 1 (links onderin) zien we hetzelfde verschijnsel. De grafiek geeft vrij simpel weer dat de afstand stijgt (niet daalt) als we het feedernummer verhogen (2, 3, 4, 5…) en ook als we het feedernummer verlagen (180,179, 178, …).



Deze monotonie-eigenschap geldt voor alle feederposities. Die monotonie wordt veroorzaakt door twee zaken:

– de verhouding van de rechthoek van de omtrek van de machine is kleiner dan 2/1. Als we een lange platte rechthoek nemen van 100:1 en we nemen een feeder in het midden van de lange zijde,

is de monotonie verdweenen. Het midden van de overzijde is dan dichterbij dan de twee overliggende hoekpunten.
– het gebruik van de supremum-norm, de Euclidesnorm geeft dit resultaat zeker niet.



Hierboven zien we een voorbeeld van het niet monotoon zijn bij de Euclidesnorm. De afstand naar feeder 31 is niet monotoon, zoals we simpel zien aan de boog in het midden. Conclusie: het zijn juist de specifieke eigenschappen van de MP200-plaatsingsmachine die deze eigenschap mogelijk maken.

Maar wat is nu het nut van deze analyse? We kunnen bij de berekening nu het volgende algoritme gebruiken: beide tools hebben een lijst met feeders, op volgorde van de nummering. We doorlopen de lijst van tool 0. We bekijken bij alle feeders voor tool 0 maar twee feeders van tool 1: degene die er vlak achter ligt en degene die er vlak voor ligt. Daarvan berekenen we de afstand, en vergelijken die met de kleinste afstand tot nu toe. Het algoritme is dan $O(2N)$ in plaats van $O(N^2)$. En dat is een flinke winst.

Conclusie

De doos met puzzels is weer dicht. De deksel is even een beetje opgelicht geweest. Maar we geven nog even een probleem voor thuis. In de Euclidesnorm kennen we de gelijkzijdige driehoek: drie punten hebben gelijke afstand. In de supremum-norm: hoeveel punten kunnen er gelijke afstand hebben tot elkaar?

Plaatsingsmachines vormen de aanleiding tot dit artikel. De puzzels kwamen voort uit verschillende onderdelen, vooral van het plaatsingsproces en dan vooral de optimalisatie daarvan. De TSP komt hierbij ook voor, op verschillende plaatsen. De meest verrassende conclusie is wel: de supremum-norm leeft ook in de praktijk.

Wat ik heb willen overbrengen, is het volgende: als je ergens een machine openmaakt, dan zie je onmiddellijk een grote verzameling puzzels. Die conclusie wil ik graag extrapoleren naar een variant van de uitspraak in *Forrest Gump*:

Het leven is een doos met puzzels.

Jan Verbakel,
wiskundige, maar vooral puzzelaar