RIJK WORDEN MET WISKUNDE

+
35 manieren waarop formules je leven (kunnen) veranderen

Richard Elwes

ONDERWIJSPROBLEMEN

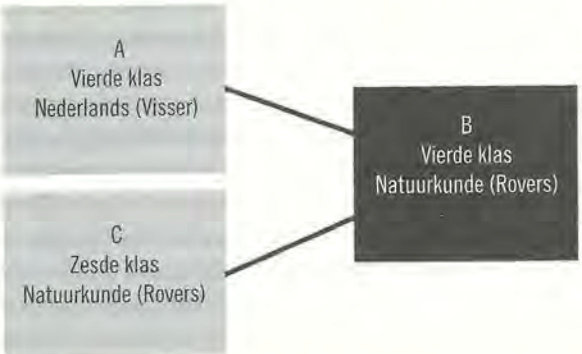
De lastige wereld van roosters

Op de juiste tijd op de juiste plek zijn, wordt in het algemeen beschouwd als een fortuinlijke gang van zaken. Dat kan de gelukkige doorbraak betekenen die een schitterende carrière inluidt. Op een meer werelds en literair niveau – hoewel net zo handig voor carrièreperspectieven – is er de dagelijkse routine voor schoolkinderen, universiteitsstudenten en hun onderwijzers en hoogleraren. Die ligt besloten in het bescheiden lesrooster. Het wordt als vanzelfsprekend aangenomen, maar dat eenvoudige rooster met lesuren en activiteiten verbergt een boel ingewikkeldheden – horden die moeten worden genomen door de administrateur belast met het samenstellen ervan, door te balanceren met vakgebieden, jaren, klassen, leraren (fulltime en in deeltijd), lokalen, uitrusting, budget, buitenschoolse activiteiten en ga zo maar door – probeer daarbij maar het hoofd koel te houden. De taak zit vol moeilijkheden, en de uitdagingen gaan zo ver dat er gigantische getallen ontstaan, en leiden naar de grootste vraag in theoretische computerwetenschappen.

In termen van wiskundige genres is roosteren een voorbeeld van een indelingstaak, een taak die bijvoorbeeld voorkomt bij transportnetwerken, in fabrieken en in organisaties en educatieve instellingen. Bij het bouwen van een auto bijvoorbeeld kunnen sommige taken parallel worden uitgevoerd, zoals het assembleren van de motor en de portieren, terwijl andere in een bepaalde volgorde moeten worden gedaan. Het is bijvoorbeeld geen goed idee om de wielen te bevestigen voordat de aandrijfstang is aangebracht. Die waarneming inspireerde Henry Ford toen hij de 'lopende band' bedacht, een productielijn waarbij opeenvolgende werknemers een handeling aan het product verrichten. Het lesrooster op een school kan conceptueel een van de gemakkelijkere indelingstaken lijken, omdat het alleen maar bestaat uit het vermijden van overlappen in plaats van het doen van dingen in de juiste volgorde. Dat laat zich gevoelsmatig echter niet vertalen naar een snelle manier om het te voltooien.

Voor een wiskundige kan de constructie van een tijdschema worden vertaald naar een schijnbaar totaal andere terminologie van netwerken. We beginnen met een lijst van uit te voeren taken – misschien de lessen die op een enkele dag moeten worden gegeven, zeg derde klas natuurkunde, vierde klas natuurkunde, vierde klas Nederlands, enzovoort – en elke taak wordt op een pagina met een knooppunt aangegeven. Het probleem is dat bepaalde taken niet tegelijkertijd kunnen worden uitgevoerd. Dat blijkt doordat taken aan elkaar grenzen of met een lijn zijn verbonden. Het knooppunt van Nederlands in de vierde klas, gegeven door meneer Visser, zal zijn verbonden met het knooppunt van de natuurkundeles van de vierde klas, gegeven door mevrouw Rovers, en dat is weer verbonden met (omdat het daarmee botst) de natuurkundeles van de zesde klas, omdat die les ook door mevrouw Rovers wordt gegeven. Dit voorbeeld bestaat tot zover uit drie knooppunten waarbij A is verbonden met B dat weer is verbonden met C, maar C is niet verbonden met A.

Als een dergelijk schema van knooppunten en verbindingslijnen is opgesteld, wordt het tijd voor inkleuren. We geven aan elk knooppunt een kleur en we kiezen de kleuren zo slim dat geen enkele lijn twee knooppunten met dezelfde kleur verbindt. Het idee is dat de kleuren verschillende tijdsperioden in het tijdschema voorstellen. Rode knooppunten kunnen staan voor de les van 9 uur tot 10 uur in de ochtend, terwijl groen staat voor 10 uur tot 11 uur in de ochtend. Zolang we geen knooppunten met dezelfde kleur verbinden, vermijden we tijdsoverlappen. We zijn ook aangekomen bij een belangrijke vraag: wat is het kleinste aantal benodigde kleuren? Voor wiskundigen is het minimale aantal kleuren het chromatische getal van het netwerk. Voor onze school stelt het een fundamenteel stuk informatie voor: het minimale aantal benodigde tijdsloten voor het voltooien van alle taken. Als het



Dit kleine netwerk met drie knooppunten heeft een chromatisch getal van 2, omdat A en C elk verbonden zijn met B, maar niet met elkaar. Bij grotere netwerken is het moeilijk deze waarde te bepalen.

chromatische getal groter is dan het aantal perioden dat beschikbaar is op een schooldag, dan is de betreffende tijdschemataak onmogelijk! Aangenomen dat dat niet het geval is, kunnen er als het chromatische getal is bepaald verscheidene manieren zijn om het inkleuren te voltooien – er zijn vele, even goede uitkomsten van het tijdschema.

Een hemelrijk aan berekeningen

Tot nog toe wijst al dat gedoe met knooppunten en kleuren, op een verstrooiende afleiding met gebruik van kleurpotloden, pennen en linialen en misschien de belofte van een plotselinge openbaring als alle verbanden op hun plaats vallen. Helaas gaat dat niet zo. Wiskundigen hebben de afgelopen decennia dergelijke taken uitvoerig bestudeerd en wat ze vonden is slecht nieuws voor wie een lesrooster voor de school opstelt. Het vinden van het chromatische getal van een netwerk wordt in vaktermen aangeduid als een '*NP*-volledig probleem', en dat houdt in dat het moeilijk is en veel tijd vergt – en in het algemeen is het onwaarschijnlijk dat er een snelle sluiproute is.

Natuurlijk zijn kleurpotloden en linialen al jaren geleden onder in de archiefkast opgeborgen en in dit computertijdperk mag men zich vervolgens afvragen hoe moeilijk 'moeilijk' is en hoelang 'lang' duurt. De antwoorden op deze vragen zullen toch zeker afhangen van de gebruikte software, de snelheid en rekenkracht van de computer en hoe vernuftig de gebruikte methode is? Dat is allemaal waar, maar er is een meer diepgaand inzicht in de moeilijkheid van een probleem, dat volledig objectief is en zelfs grotendeels immuun voor technologische vooruitgang. Dat valt in de categorie van computationele complexiteit, een onderwerp dat zowel in universitaire wiskundefaculteiten als in de onderzoekslaboratoria van softwarebedrijven wordt onderzocht. Het idee is om de moeilijkheid van een probleem te classificeren naar het minimale aantal benodigde stappen om die te voltooien – hoe minder stappen, des te gemakkelijker het probleem is.

De grootte van het netwerk is duidelijk van belang, en de cruciale vraag is hoe snel de berekening groeit in verhouding tot die omvang. Tot dusverre wijst onderzoek uit dat voor een gewoon netwerk met n knooppunten er 2^n stappen (dat is n-maal 2-en met elkaar vermenigvuldigd) nodig zijn om het chromatische getal te vinden. Dat is nogal intimiderend nieuws, want die uitdrukking groeit inderdaad zeer snel. Als $n = 10$ betekent het dat onze berekening $2^{10} = 1024$ stappen nodig heeft; verdubbeling van de netwerkgrootte tot 20 knooppunten vereist een berekening van $2^{20} = 1.048.576$ stappen terwijl een netwerk met 30 knooppunten 2^{30} stappen nodig heeft, en dat is meer dan een miljard. Duidelijk lopen deze berekeningen gierend uit de klauwen. Op die manier vergt een netwerk met 90 knooppunten (of een tijdschema met 90 verschillende lessen) een berekening die langer duurt dan de leeftijd van het heelal, zelfs als er elke nanoseconde een stap wordt berekend. Dat toont de verwoestende kracht van exponentiële groei.

P of *NP* – de polynomiale vraag

Gelukkig is nog niet alles verloren. De meest praktische computerprogramma's gebruiken een veel vriendelijkere klasse van algoritmen, met polynomiale groei. Om dat te illustreren, kunnen we ons voorstellen dat er een gloednieuwe manier van netwerkanalyse is ontdekt die n^2 stappen vereist voor berekening van het chromatische getal van een netwerk met n knooppunten. Voor een netwerk met 10 knooppunten heeft het programma $10 \times 10 = 100$ stappen nodig. Tegen de tijd dat $n = 90$ bereiken we een aantal van $90 \times 90 = 8100$ stappen. De meesten zullen het er mee eens zijn dat dit een aanzienlijke verbetering is ten opzichte van de leeftijd van het heelal: een moderne computerprocessor doet dat in een oogwenk.

Voor het geoefende oog blijkt de snelheid van deze twee algoritmen uit de tegengestelde algebraïsche uitdrukkingen – de (exponentiële) 2^n en de polynomiale n^2. Andere voorbeelden van polynomen zijn uitdrukkingen zoals n^3, n^4, n^5. Deze algebraïsche labels maken het mogelijk om problemen naar moeilijkheid te classificeren. Bekend is dat er veel problemen zijn die *kunnen* worden opgelost in polynomiale tijd. De verzameling van al die polynomiaal oplosbare problemen staat traditioneel bekend als *P*. Een handige (maar iets versimpelde) vuistregel is dat *P* bestaat uit alle problemen waarvoor het mogelijk is om een algoritme te bedenken dat snel genoeg is om werkelijk nuttig te zijn.

Wat een uitkomst zou zijn voor onze oorspronkelijk lesroosteruitdaging is een algoritme dat de klus snel klaart, dus in polynomiale tijd. Bestaat dat? De theoretische vraag luidt hier of het inroosteren wel of niet valt in de polynomiale categorie – bevindt het zich 'in *P*'? Het slechte nieuws is dat het antwoord, voor zover we dat weten, 'nee' is.

Er is echter een nauwe verwant van de lesroostertaak die snel kan worden uitgevoerd, namelijk een antwoord nagaan. Als we een netwerk met 90 knooppunten beschouwen dat overeenkomt met ons lesroosterprobleem en we krijgen een schema waarin dat met acht kleuren is ingekleurd, dan kunnen we vrij snel nagaan of de geboden oplossing voldoet. We moeten slechts van iedere verbindingslijn nagaan of die geen twee knooppunten met dezelfde kleur verbindt. Van dergelijke problemen, die in polynomiale tijd kunnen worden uitgevoerd, zegt men dat ze 'in *NP*' zijn (*NP* staat voor 'niet-deterministische polynomiale tijd').

En daar schuilt een enorm addertje onder het gras. Het grootste raadsel in de theoretische computerwetenschappen is of elk probleem dat 'in *NP*' is wel of niet ook 'in *P*' moet zijn. Op het eerste gezicht luidt het voor de hand liggende antwoord: 'nee'. En het is waar dat de meerderheid van de computerwetenschappers inderdaad gelooft dat op die manier *P* niet kan overeenkomen met *NP*. Er is wel gezegd dat als de wiskundigen dezelfde standaarden voor bewijsvoering als bij andere vakgebieden zouden hanteren, ze dat al lang geleden als een universele wet zouden hebben aanvaard. Gedurende de halve eeuw nadat de wiskundige van Oostenrijkse komaf Kurt Gödel die vraag opperde, hebben veel getalenteerde geleerden die zeer diepgaand onderzocht. Met name waren dat Stephen Cook, Leonid Levin en Richard Karp, die met hun analyse begin jaren zeventig deze vraag de status van de belangrijkste vraag in de gehele wiskunde verschaften. De vrucht van hun onderzoek is dat er bepaalde problemen zijn – waaronder het maken van een tijdschema – die in *NP* zijn maar niet in *P*.

Wiskundigen werken echter volgens een ander, meer strikt niveau van discipline. Het volstaat niet om waar te nemen dat het erop *lijkt* dat het maken van een tijdschema zich niet in *P* bevindt. Er is een keihard bewijs nodig dat het niet zo is. Dat is er nog niet, noch voor het maken van tijdschema's nog voor andere *NP*-problemen.

In feite vormt tijdschema's indelen een speciaal geval. Niet alleen is het in *NP*, maar het is *NP*-volledig. Dat betekent dat het van alle *NP*-taken een maximale moeilijkheid heeft en als een *NP*-probleem buiten *P* ligt (ofwel definitief niet polynomiaal berekenbaar is) dan geldt dat ook voor het indelen van tijdschema's. Omgekeerd verschaft dit het bescheiden opstellen van lesroosters op scholen een onverwachte status: als iemand een gemakkelijke manier ontdekt om dat voor elkaar te krijgen, in volledige algemeenheid, dan zal dat worden gezien als een van de grootste en meest onverwachte prestaties in de moderne wiskunde, aangezien daaruit automatisch volgt dat hij of zij is gestuit op een bewijs voor $P = NP$.

Gewoon uitproberen – een wiskundig alternatief

Wie zich nu afvraagt hoe het kan dat scholen op deze planeet kunnen functioneren terwijl het maken van een lesrooster een wiskundig raadsel van dergelijke monsterlijke proporties is, wordt dat vergeven. Zeker zullen er maar weinigen zijn onder hen die zijn belast met het samenstellen van een rooster, die zich willen meten met de verschrikkelijke tweeling exponentiële groei en *NP*-volledigheid. De waarheid is dat computers alternatieven bieden die de afzonderlijke gevallen van het probleem beheersbaar maken.

Allereerst is het in feite niet noodzakelijk om het chromatische getal van een graaf (zoals een dergelijk probleem wordt genoemd) te vinden. Er zijn immers een vast aantal lesuren tijdens een schooldag, bijvoorbeeld zeven. Er schuilt geen voordeel in het persen van het volledige rooster in zes lesuren, zelfs als dat mogelijk blijkt, dus we hebben niet het beste antwoord nodig, maar gewoon een dat voldoet. Ten tweede geldt *NP*-volledigheid voor het probleem in volledige algemeenheid –

een methode voor het vinden van het chromatische getal van *elk netwerk hoe dan ook*. In bepaalde gevallen kunnen er regelmatigheden in een netwerk zijn die het probleem gemakkelijker oplosbaar maken. Zo kan het zijn dat elke klas een ruwweg vergelijkbaar aantal lessen heeft en dat hetzelfde geldt voor leraren. Dat zorgt voor een soort symmetrie in het netwerk die van pas komt – met een techniek die bekendstaat als een genetisch algoritme.

De afgelopen jaren zijn genetische algoritmen populaire gereedschappen geworden. Ze zijn ontstaan bij het onderzoek naar kunstmatige intelligentie (zie *Elektronische hersenen scheppen*). Ze blijven niet rondzoeken naar het beste antwoord zoals een traditioneel computerprogramma doet. In plaats daarvan doen ze een gok en proberen ze die dan aan te passen en te verfijnen, zodat ze geleidelijk hun weg 'voelen' naar een werkbare oplossing. Een dergelijk systeem zou willekeurig een paar knooppunten kunnen inkleuren en daarna proberen om die gekleurde gebieden uit te breiden, overlap te voorkomen, veranderingen te maken, of zo nodig een stap terug te doen. De aanduiding 'genetisch algoritme' komt voort uit een analogie met evolutie, die plaatsvindt volgens de twee processen van willekeurige mutatie ('de gok') gevolgd door natuurlijke selectie ('de verfijning'). Hun benadering komt treffend overeen met hoe een mens het probleem zou aanpakken, hoewel de computer het voordeel heeft van grotere snelheid – en bovendien nooit verveeld of gefrustreerd raakt. Op merkwaardige wijze zijn we bijna weer teruggekeerd bij onze kleurpotloden en linialen, gewoon dingen uitproberen.

Tegenwoordig worden genetische algoritmen toegepast in allerlei hypermoderne toepassingen, van het kraken van codes tot kunstmatige creativiteit – het produceren van originele kunst of muziek door de computer. Tot de aantrekkelijke voorbeelden van het eenentwintigste-eeuwse technologische vermogen behoort zeker ook het schijnbaar saaie lesroosterprobleem, een bedrieglijk alledaags probleem, met werkelijk opmerkelijke verscholen diepgang.