Rob Bisseling
Mathematisch Instituut
Universiteit Utrecht
Postbus 80010
3508 TA Utrecht
R.H.Bisseling@uu.nl

Oratie

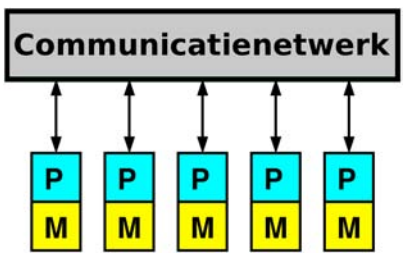
Parallel, groen en

Computers met meerdere processorkernen zijn doorgedrongen in ons dagelijks leven: moderne desktop pc's en draagbare computers zijn vaak voorzien van dual-core of soms zelfs quad-core processoren. Hoe moeten we onze software en rekenmethoden aanpassen zodat we deze parallelle processorkracht ook daadwerkelijk kunnen gebruiken? Hoe kunnen we tegelijkertijd meerdere bewerkingen op deze computers uitvoeren? In zijn oratie, uitgesproken op 8 december 2009 aan de Universiteit Utrecht, zet Rob Bisseling, profileringshoogleraar scientific computing, uiteen hoe we parallellisme kunnen inzetten in diverse toepassingen, waarom dit een groene technologie is en hoe snellere berekeningen van maatschappelijk nut kunnen zijn.

Het is parallel, groen en snel. Dit klinkt als een raadsel en zo is het ook bedoeld. In de volgende tekst zal ik dit raadsel ophelderen en u laten zien hoe deze woorden mijn vakgebied, *scientific computing*, karakteriseren.

Parallel

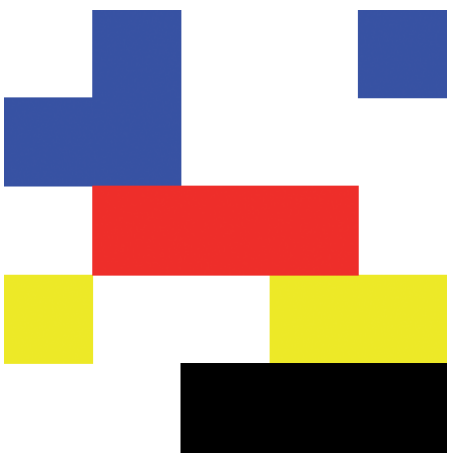
U rijdt misschien wel eens over de parallelweg, evenwijdig aan de snelweg. Of u droomt van een parallel universum, of wat bescheidener van een parallelle wereld, evenwijdig aan de echte, maar net iets anders. Zonder dat u het beseft gebruikt u waarschijnlijk al een parallelle computer, en dat is de betekenis van het woord 'parallel' waar het hier om gaat. Als ík droom over iets parallels dan gaat het om parallelle algoritmen: rekenrecepten om parallelle computers te gebruiken.



Figuur 1 Parallelle computer met 5 processoren (P), ieder met een eigen geheugen (M, *memory*). De processoren communiceren met elkaar via een netwerk.

Wat is een parallelle computer? Simpel gezegd, een computer die tegelijkertijd, dus parallel, verschillende operaties uitvoert. Daarvoor beschikt die computer over verschillende processoren, zie Figuur 1. Een computer met 1000 processoren kan in principe 1000 maal zo snel rekenen als met een niet-parallelle, sequentiële computer. Dat is veelbelovend, en vormt op zich al een mooie motivatie voor fundamenteel onderzoek naar parallellisme. Mijn eigen interesse in dit vakgebied werd gewekt door een cursus *Parallel and distributed algorithms* van Larry Rudolph aan de Hebreeuwse universiteit van Jeruzalem, waar ik mijn promotieonderzoek deed van 1982 tot 1986. Parallelle computers waren toen vooral van theoretisch belang, en pas in 1986 kwam de eerste commerciële machine, een Intel hypercube, op de markt. Ik was zo fortuinlijk vanaf 1987 op het researchlaboratorium van Shell in Amsterdam te werken, waar men al snel een parallelle computer met wel 400 processoren aanschafte van Europees fabrikaat, gebaseerd op een computerchip genaamd de *transputer*. De goede connecties van de chipfabrikant (Inmos in Bristol) en het succes van deze chip zorgden er voor dat de term 'transputer' al snel een trefwoord in de *Concise Oxford English Dictionary* werd.

We maken een sprong in de tijd, en belanden in 2009. Mijn broer wil een nieuwe thuiscomputer kopen en laat mij een folder zien van de lokale pc-verkoper ergens in de Achterhoek, onze geboortestreek. De keuze is tussen een single-core, dual-core, en een quad-core computer, ter plekke in elkaar te zetten op basis van losse componenten, zo van het schap. Als u weet dat een *core* een rekenkern is, en er meerdere kernen op een



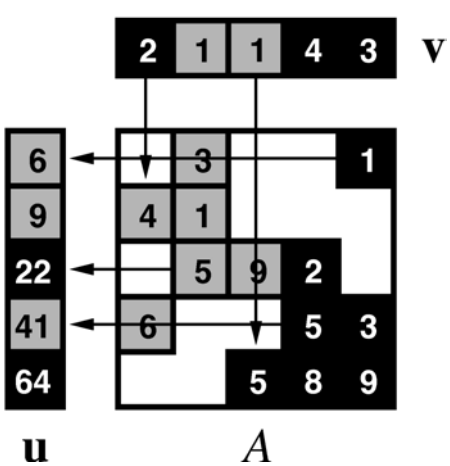
Figuur 2 Matrix met 13 niet-nul elementen verdeeld over vier processoren door het programma Mondriaan [21]. De blauwe processor heeft vier niet-nul elementen, de anderen drie.

Rob Bisseling

snel

computerchip geplaatst kunnen worden, dan is de eigenlijke vraag, hoeveel parallellisme wil je hebben in de computer: 1-, 2-, of 4-voudig? Hierbij is enkelvoudig natuurlijk sequentieel, en spreken we pas vanaf 2 kernen over echt parallellisme. Ruwweg betaal je zo'n 100 Euro meer voor elke extra kern.

Meer is kennelijk beter, maar let op: kan de computer de verschillende kernen ook echt tegelijk gebruiken? Eén kern kan een bepaalde nuttige taak verrichten, en voor een wiskundige is dat vaak een berekening of de visualisatie ervan, en een andere kern zou de elektronische post af kunnen handelen. Wat kunnen we 4 kernen echter laten doen? Om die goed te gebruiken, moeten we onze software geschikt maken. Dit betekent dat we de onderliggende algoritmen, de rekenrecepten, moeten *parallelliseren*. Dit is een uitdaging, voor grote bedrijven als chipmaker Intel, computerfabrikant Apple en softwareproducent Microsoft, en binnen het wiskundig gebied voor de makers van Matlab en Mathematica.



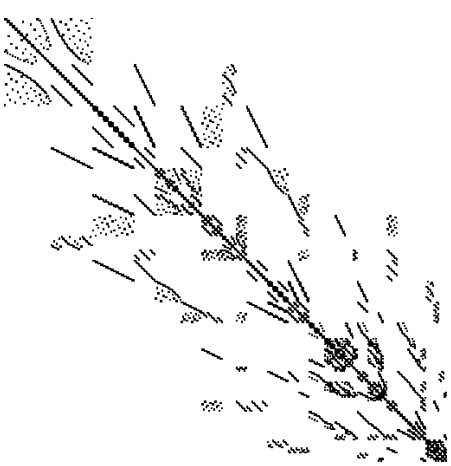
Figuur 3 Twee processoren, zwart en grijs, communiceren met elkaar tijdens de parallelle berekening van het matrix–vector product $\mathbf{u} = A\mathbf{v}$ [2].

Een intellectuele uitdaging ook voor numeriek wiskundigen, die wiskundige problemen oplossen op de computer, vaak via benaderingen en snelle algoritmen. Sinds 1993 geef ik aan deze universiteit met veel plezier het vak Parallelle Algoritmen, waarin we allerlei basisberekeningen parallelliseren volgens twee principes:

1. *Verdeel en heers*: verdeel de gegevens en het rekenwerk eerlijk over de processoren, zodat niemand hoeft te wachten en niemand te veel werk heeft, zie Figuur 2.
2. *Communiceer met mate*: zo min mogelijk, maar wel voldoende. Om samen een probleem op te lossen is communicatie nodig: gegevens worden verstuurd tussen de processoren of kernen, zie Figuur 3. Een informatieve boodschap kan een processor weer aan het werk zetten die anders niet verder zou kunnen met zijn berekening. Te veel communicatie kost echter tijd en houdt de processoren maar van het werk. Het versturen van de informatie kost zelf ook tijd, vaak meer dan het bewerken ervan, en dit willen we dus vermijden.

Een matrix is een soort schaakbord gevuld met gegevens, de elementen. Een ijle matrix, zie Figuur 4, bevat weinig elementen ongelijk nul, zoals ijle lucht weinig zuurstof bevat.

Niet-nullen zijn de zuurstof van onze berekeningen. Gegevens die een numerieke waarde ongelijk nul bevatten worden door ons gemanipuleerd net zo lang tot we betekenisvolle uitkomsten krijgen. Gegevens die gelijk aan nul zijn kunnen we overslaan. In de scientific computing is een matrix met een miljoen rijen en kolommen heel normaal. Vaak zijn onze matrices vierkant, maar dat hoeft niet. Voor mij worden matrices pas interessant als ze veel nullen bevatten. Je krijgt dan een structuur van nullen en niet-nullen, de *ijlheidsstructuur*. Deze structuur kan benut worden om berekeningen te versnellen. Een nul ergens bij optellen is gratis, en met nul ver-



Figuur 4 340×340 matrix cage7 met 3084 niet-nullen (2.7% van het totaal aantal matrixelementen). De matrix stelt een Markov model voor van DNA elektroforese [20]. Rijen en kolommen representeren hierbij mogelijke toestanden en matrixelementen representeren overgangswaarschijnlijkheden tussen de toestanden.

menigvuldigen levert een zekere uitkomst van nul op. Ook hoef je nullen niet op te slaan in het computergeheugen. Immers, het volstaat de niet-nullen en hun posities op te slaan (een niet-nul met waarde 7.5 in rij 6 en kolom 12, of bij schaken een paard in rij 1 en kolom g). Deze geheugenbesparing zorgt ervoor dat je veel grotere problemen kunt oplossen.

In het deelgebied *Combinatorial scientific computing* [6] (afgekort CSC) staan ijle matrices centraal. CSC is een van mijn researchinteresses. Als onderzoeker probeer ik algoritmen te bedenken die de ijlheidspatronen van matrices benutten om matrixberekeningen sneller te maken, onder andere door deze te pa-

Mondriaanpartitionering

De Mondriaanpartitioneerder verdeelt een ijle matrix A in p stukken, ieder bestemd voor een processor (of een core). De processoren zijn genummerd 0 t/m $p-1$. Het totaal aantal niet-nullen (*nonzeros*) van de hele matrix is $nz(A)$ en het aantal van processor s is $nz(A_s)$. De eerlijke werkverdeling wordt opgelegd door de voorwaarde

$$nz(A_s) \leq (1+\epsilon)\,\frac{nz(A)}{p}, \quad \text{voor } s = 0, 1, \ldots, p-1,$$

waarbij ϵ de toegestane onbalans in de werkverdeling is. Het *communicatievolume* behorend bij de verdeling is

$$V(A_0, A_1, \ldots, A_{p-1}) = \sum_i (\lambda_i - 1) + \sum_j (\mu_j - 1),$$

waarbij λ_i het aantal processoren is dat een niet-nul in matrixrij i heeft en μ_j het aantal processoren dat een niet-nul in matrixkolom j heeft. Deze metriek geeft weer dat een processor informatie moet sturen naar alle andere processoren die een deel van de rij of kolom bezitten. Er geldt dat $1 \leq \lambda_i, \mu_j \leq p$, voor alle i, j. (We nemen aan dat geen enkele rij of kolom leeg is.)

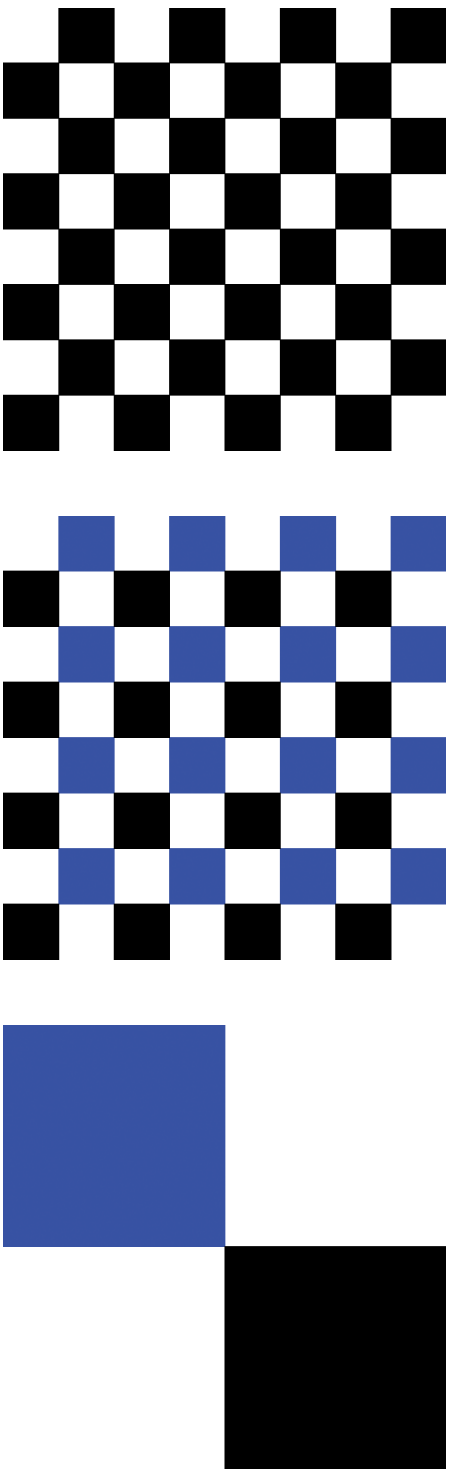
Het partitioneren gebeurt via *orthogonale recursieve bisectie*, het herhaaldelijk in tweeën splitsen van de matrix waarbij zowel de rijrichting als de kolomrichting wordt geprobeerd en de beste richting (met het kleinste volume) wordt gekozen. Dit wordt weergegeven door het onderstaande recursieve algoritme.

De functie roept zichzelf aan, met de helft van het oorspronkelijke aantal processoren. De waarde van ϵ wordt hierbij aangepast: het matrixdeel met minder niet-nullen krijgt een grotere ϵ voor de resterende partitionering. De splitsing in tweeën gaat er vanuit dat alle toekomstige splitsingen dezelfde onbalans δ creëren, resulterend in een uiteindelijke groeifactor $1+\epsilon = (1+\delta)^q \approx 1 + q\delta$, wat leidt tot de optimale keuze $\delta = \epsilon/q = \epsilon/\log_2 p$ voor de eerste splitsing.

De splitsing in twee stukken gebeurt via een *multilevelmethode*, waarbij (in het geval van de kolomrichting) eerst matrixkolommen die op elkaar lijken via een herhaalde matchingsprocedure worden samengevoegd, daarna de ontstane superkolommen worden verdeeld in twee groepen met behulp van het Kernighan–Lin [9] algoritme, en de resulterende verdeling uiteindelijk wordt verfijnd op alle verschillende niveaus van de multilevelmethode.

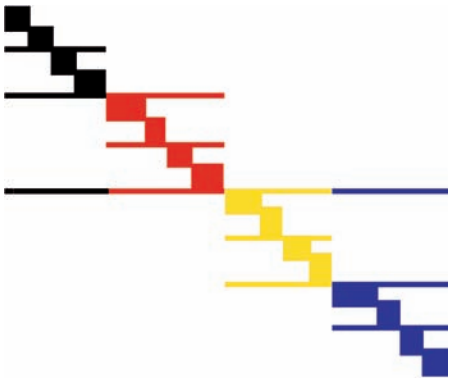
aanroep: $(A_0, \ldots, A_{p-1}) :=$ **MatrixPartition**(A, ϵ, p).
invoer: A: ijle matrix,
p: aantal processoren, $p = 2^q$ met $q \geq 0$,
ϵ: toegestane onbalans, $\epsilon > 0$.
uitvoer: $(A_0, \ldots, A_{p-1})$: partitionering van A.

if $p > 1$ **then**
  {Splits in 2 delen}
  $(B_0^{\text{rij}}, B_1^{\text{rij}}) := split(A, \text{rij}, \frac{\epsilon}{q})$;
  $(B_0^{\text{kol}}, B_1^{\text{kol}}) := split(A, \text{kol}, \frac{\epsilon}{q})$;
  if $V(B_0^{\text{rij}}, B_1^{\text{rij}}) \leq V(B_0^{\text{kol}}, B_1^{\text{kol}})$
    then $(B_0, B_1) := (B_0^{\text{rij}}, B_1^{\text{rij}})$;
    else $(B_0, B_1) := (B_0^{\text{kol}}, B_1^{\text{kol}})$;
  {Splits verder}
  $maxnz := (1+\epsilon)\frac{nz(A)}{p}$;
  $\epsilon_0 := \frac{maxnz}{nz(B_0)} \cdot \frac{p}{2} - 1$;
  $\epsilon_1 := \frac{maxnz}{nz(B_1)} \cdot \frac{p}{2} - 1$;
  $(A_0, \ldots, A_{p/2-1}) :=$ **MatrixPartition**$(B_0, \epsilon_0, \frac{p}{2})$;
  $(A_{p/2}, \ldots, A_{p-1}) :=$ **MatrixPartition**$(B_1, \epsilon_1, \frac{p}{2})$;
else $A_0 := A$;

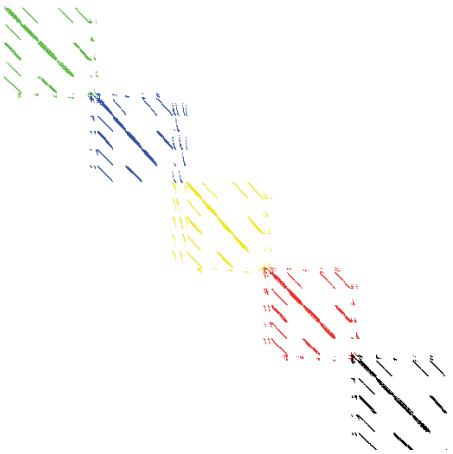


Figuur 5 8×8 matrix chess8 met 32 niet-nullen.
(a) originele matrix;
(b) matrix eerlijk verdeeld over 2 processoren, waarbij alle niet-nullen in een rij of kolom dezelfde kleur (processor) hebben gekregen;
(c) dezelfde matrixverdeling, maar nu na rijpermutatie (blauwe rijen eerst) en kolompermutatie (blauwe kolommen eerst). Bij deze verdeling is geen communicatie nodig tijdens parallelle matrix–vector vermenigvuldiging.

Figuur 6 Piet Mondriaan: *Compositie nr. 2 in Lijn en Kleur*, 1913. Gemeentemuseum Den Haag



Figuur 7 *Cache-oblivious* permutatie van rijen en kolommen. Gemengde rijen worden in het midden geplaatst, recursief, om een geleidelijke overgang in cachegebruik van gegevens aan de linkerkant naar gegevens aan de rechterkant te bewerkstelligen. Dezelfde permutatie kan gebruikt worden voor elke cache-architectuur, ongeacht de precieze cache-groottes en cache-hiërarchie [22].



Figuur 8 Matrix `lns3937` met 3937 rijen en kolommen, en 25407 niet-nullen. De matrix komt voort uit de oplossing van de Navier–Stokes vergelijkingen voor vloeistofstroming en is door de Mondriaanpartitioneerder in 5 gelijke delen gesplitst, waarbij 3% onbalans is toegestaan in het aantal niet-nullen per processor. In dit plaatje is de matrix gepermuteerd naar de bijbehorende SBD-structuur. Een filmpje van de partitionering gemaakt met MondriaanMovie van Bas Fagginger Auer is te zien op www.math.uu.nl/people/bisseling/oratiefilmpje.avi

rallelliseren. Van 1987–1993 heb ik bij Shell samen met collega's Daniël Loyens en Hans van de Vorst een hele parallelle lineaire algebra bibliotheek ontwikkeld, PARPACK, voor berekeningen met ijle en ook volle matrices, gericht op raffinaderijoptimalisatie. De matrices verdeelden we toen over de processoren via twee-dimensionale cyclische verdelingen.

Later, in Utrecht, ben ik dieper ingegaan op het verdelingsprobleem, ook wel *partitionering* genoemd. Een voorbeeld hiervan ziet u in Figuur 5. Ik mocht daarbij profiteren van mijn interactie met vele creatieve, slimme en kritische Utrechtse studenten die mijn woord nooit meteen voor waar aannamen. Als ik dacht dat Cartesische verdelingen het summum waren, dan liet Brendan Vastenhouw mij zien dat voor matrix-vectorvermenigvuldiging niet-Cartesische verdelingen soms beter zijn; daar is later de Mondriaan-verdeling uit geboren. Het is een genot elk jaar weer dergelijke studenten tegen te komen, deze te mogen begeleiden en samen onderzoek te doen.

Piet Mondriaan is een groot Nederlands schilder, geboren in Amersfoort in 1872, overleden in New York in 1944. Hij is zijn loopbaan figuratief begonnen, met het schilderen van onder andere de molens en bomen aan de rivier het Gein. De bomen werden steeds rechthoekiger en abstracter, en de kleuren steeds meer primair. Het groen verdween. Zijn schilderijen zijn een inspiratiebron voor velen, en ook voor mij. In 2001, toen Brendan en ik een naam zochten voor onze verdelingssoftware voor matrices kwamen we uit op Mondriaan. We hebben de naam bijna een jaar geheim gehouden om het achterliggende verdelingsidee niet te verklappen vóór publicatie. In mei 2002 zetten we versie 1.0 op het web en stuurden ons artikel naar SIAM Review, waar het uiteindelijk verscheen in 2005 [21].

Versie 2.0 brachten we in juli 2008 uit op onze website, met Figuur 6 als begeleidende foto. Deze versie bevat geavanceerde methoden voor de verdeling van vectoren, naast nieuwe, fijnkorreliger en hybride methoden voor matrices. Extra aandacht hebben we hierbij besteed aan de kwaliteit van de software via uitvoerige tests, zowel van afzonderlijke modules, als stresstests van het hele systeem. Wij stellen onze software als *open source* beschikbaar. Iedereen kan elke regel programmacode lezen, veranderen naar eigen behoefte en fouten opsporen. Mijn opvatting is dat software zelf een steeds belangrijkere publicatie wordt, naast de traditionele artikelen in wetenschappelijke tijdschriften. Software als eindproduct van onderzoek wordt nog wel eens ondergewaardeerd. Het is goed te beseffen dat het gebruik van ons onderzoek en wellicht ook het aantal citaties van onze artikelen sterk beïnvloed worden door de beschikbaarheid van goed gedocumenteerde en robuuste software.

Visualisatie is belangrijk voor scientific computing. Allereerst om grote hoeveelheden gegevens snel te begrijpen, maar ook om beter nieuwe algoritmen te kunnen bedenken. De eerste plaatjes van matrices in De Stijl maakte ik met de hand, hokje voor hokje, niet-nul voor niet-nul, zoals een schilder zou doen op het canvas. Een nuttige vingeroefening, die later hielp bij het automatisch creëren van een mooie visualisatie.

Inmiddels staat Mondriaan versie 3.0 klaar. Mijn huidige promovendi Albert-Jan Yzelman en Bas Fagginger Auer hebben veel nieuws toegevoegd aan de software. Albert-Jan heeft permutatie-algoritmen bedacht om bepaalde structuren te creëren, zoals de *Separated Block Diagonal* (SBD) structuur [22], zie Figuur 7, die erg goed werkt bij moderne computers met een ingewikkelde geheugenarchitectuur, voorzien van meerdere lagen van cache-geheugen, met prozaïsche namen zoals L1, L2, en L3 cache. Bas, promovendus sinds september 2009, heeft interfaces gemaakt voor Mondriaan met de veelgebruikte pakketten Matlab en Mathematica. Ook heeft hij het mogelijk gemaakt om het verdelingswerk in uitvoering vast te leggen in filmpjes, zie Figuur 8.

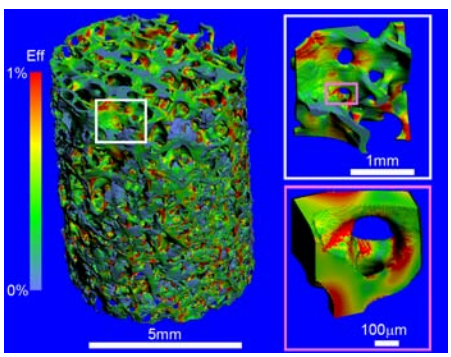
De meesten van u zullen vrijwel dagelijks een zoekmachine zoals Google gebruiken om informatie te vinden, bijvoorbeeld de locatie van het Academiegebouw in Utrecht, recente publicaties van een collega of alle betekenissen van het woord 'parallel'. Google is ontwikkeld in 1998 door twee promovendi van Stanford University, Sergey Brin en Larry Page, en bevat een fikse dosis numerieke wiskunde, zie bijvoorbeeld [10]. Brin had een TWIN-master met dubbele major wiskunde en informatica afgerond, Page een master informatica, en ze deden onderzoek binnen de afdeling informatica. De zoekmachine is begonnen uit ijdelheid: spiegeltje, spiegeltje aan de wand, wie heeft de populairste webpagina van het land? En van de hele wereld? Brin en Page vroegen zich af wie er naar hun pagina verwees op het net ontstane wereldwijde web. En natuurlijk naar alle andere pagina's, anders kun je geen vergelijking maken en dus ook geen rangorde bepalen. Maar als een pagina populair is en ook nog de door jou gezochte term bevat, dan is dat vast een goed antwoord op je zoekvraag. Hun zoekmachine draaide eerst binnen het domein van Stanford, gedoogd door een vriendelijke systeembeheerder, en groeide uit tot een groot succes. Menig wiskundige en informaticus in Stanford werd hierdoor miljonair.

Het wereldwijde web kan worden be-

Rang	Locatie	Computer	Cores	$R_{\max}$	Verbruik
1	Oak Ridge, VS	Jaguar Cray XT5	224162	1759	6951
2	Los Alamos, VS	IBM Roadrunner	122400	1042	2346
3	Univ. Tennessee, VS	Kraken Cray XT5	98928	832	
4	Jülich, Duitsland	IBM Blue Gene/P	294912	826	2268
5	Tianjin, China	Tianhe-1 NUDT	71680	563	
93	SARA, Amsterdam	IBM Power 575	3456	51	552

Tabel 1 Lijst van de snelste supercomputers ter wereld in november 2009. Voor elke computer wordt het aantal cores gegeven, de maximaal gemeten rekensnelheid in Tflop/s (10^{15} floating-point operaties per seconde) en het elektriciteitsverbruik in kWatt, exclusief koeling. Bron: http://www.top500.org

schouwd als een ijle matrix, waarbij elke rij en bijbehorende kolom een webpagina voorstelt, en elke niet-nul een link van de ene pagina naar de andere. De matrix is niet-symmetrisch, want als jij naar mij wijst, betekent dit nog niet dat ik naar jou wijs. Het bepalen van de populariteit via het aantal naar mij wijzende links komt overeen met de vermenigvuldiging van een ijle matrix met een vector, wat we zojuist met behulp van Mondriaan hebben geparallelliseerd. Ik hoop dat u bij de naam 'Mondriaan' inmiddels denkt aan het softwarepakket in plaats van de schilder! De genialiteit van Brin en Page lag erin dat ze zagen dat je de matrix herhaaldelijk met de vector moest vermenigvuldigen, zodat de populariteit van de verwijzende webpagina's meetelt bij het bepalen van de populariteit van een pagina. Een numeriek wiskundige zou dit formuleren als: wij lossen een eigenwaardeprobleem op met de powermethode. Google doet dit eens per maand voor het hele web, voor een matrix van meer dan 10 miljard rijen en 10 miljard kolommen, inderdaad meer webpagina's dan mensen op aarde, en met gemiddeld zo'n 10 niet-nullen per rij. Na 50 keer vermenigvuldigen is men bij Google uitgedanst, en is de populariteit van elke pagina bepaald. Dit is in het kort het beroemde PageRank algoritme dat ervoor zorgt dat u snel uw zoekvraag beantwoord ziet. Nu Google tot een groot bedrijf is uitgegroeid, zijn verdere ontwikkelingen van dit algoritme uiteraard bedrijfsgeheim. Immers, Yahoo! en Microsoft luisteren mee. Laat ik u verklappen dat de vindingrijkheid van onze Utrechtse studenten en promovendi niet onderdoet voor die op Stanford, en laten we hen vooral de ruimte geven. Ruimte om hun ideeën te beproeven, computers en vriendelijke programmeeromgevingen (UNIX en Linux), en ICT als spannend researchhulpmiddel en -doel, niet als routine-automatisering. Zo houden we onze getalenteerden gelukkig en krijgen we mooie resultaten, ook in Utrecht onder de Dom.



Figuur 9 Simulatie van de effectieve belasting van een menselijke ruggenwervel (uit [1]). De rekentijd is ruim 27 minuten op 8100 cores van een IBM BlueGene/L computer, waarvan 13 minuten voor het verdelen van de bijbehorende matrix door ParMetis [8] over de processoren. De matrix heeft meer dan 1,5 miljard rijen en kolommen.

Groen

In december 2009 vond in Kopenhagen de conferentie over klimaatverandering van de Verenigde Naties plaats, waarin gesproken werd over de invloed van de mens op het klimaat, en in het bijzonder de veranderingen die teweeggebracht worden door de uitstoot van CO_2, kooldioxide. Oorzaak van deze uitstoot is het gebruik van fossiele brandstoffen, als benzine in auto's, maar ook als grondstof bij de opwekking van elektriciteit nodig voor onze wasmachines, airconditioning, en, inderdaad, computers.

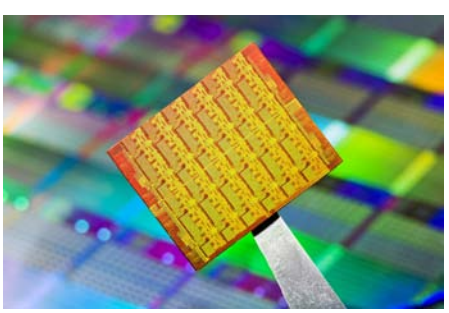
Als burger hebben wij allen een verplichting jegens de komende generaties om onze aarde in goede toestand over te dragen. Als wetenschapper hebben wij ook nog een bijzondere positie: de kennis die wij genereren kan direct benut worden om inzicht te krijgen in de processen die plaatsvinden en deze wellicht ten goede te veranderen. Als wiskundigen kunnen wij hieraan volop meedoen.

In 2007 was computergebruik al verantwoordelijk voor meer dan 2% van de totale CO_2-uitstoot. Deze zogeheten *voetafdruk* van de computergebruiker heeft in de afgelopen decennia ongemerkt kunnen groeien. Twee ontwikkelingen stuwen deze toename: het groeiend aantal computergebruikers (iedere wereldburger zijn eigen pc) en de stijgende energieconsumptie van de computerchip. Heel lang was het elektriciteitsverbruik van de chip verwaarloosbaar, en viel in het niet bij die van bijvoorbeeld de monitor, of de rondwentelende harde schijf. Het schaalgedrag als functie van de kloksnelheid is echter niet gunstig: een hogere kloksnelheid betekent meer rekenoperaties per seconde, maar helaas *veel* meer Watt benodigde elektriciteit [15]. Een chip (met 2 rekenkernen) van onze nationale supercomputer Huygens verbruikt ongeveer 320 Watt, evenveel als 3 verboden gloeilampen samen. Een chip in een pc haalt makkelijk 100 Watt, en meer energieverbruik met de bijbehorende warmteontwikkeling wilt u beslist niet op schoot hebben in uw laptopcomputer.

Het elektriciteitsverbruik bij mij thuis (3 personen en 3 computers) was 6137 kiloWattuur in 2008. Dat is omgerekend dag in, dag uit 700 Watt. De snelste supercomputer ter wereld, de Jaguar Cray XT in het Oak Ridge National Laboratory in de VS, verbruikt zo'n 7 MegaWatt, zie Tabel 1, dus 10.000 maal zoveel als mijn huishouden. Hier is werk te doen.

Ziet u in mijn verhaal vooral geen pleidooi om supercomputers uit te zetten en pc's af te danken. Ons moderne leven is te afhankelijk van de computer, en de prachtige simulaties op supercomputers leveren veel nieuwe kennis op. Simulaties van menselijke botstructuren zoals collega's Peter Arbenz en Costas Bekas op massief parallelle supercomputers uitvoeren [1], helpen bij het lokaliseren van zwakke plekken en kunnen in de toekomst fracturen helpen voorkomen en osteoporose helpen bestrijden, zie Figuur 9. De simulaties die collega Gerard Barkema en ik van amorf silicium draaien [17] kunnen ons uiteindelijk helpen betere beeldschermen en zonnecellen te produceren. De oceaansimulaties van collega Henk Dijkstra van het IMAU, een grote rekenaar op Huygens, helpen ons te voorspellen of de warme golfstroom over precies 69 jaar nog bestaat. Daarvan willen we blijven profiteren. Ons doel is dit te doen via *green computing*, groen rekenen.

We moeten de technologie gebruiken om



Figuur 10 Intel Single-Chip Cloud computer met 48 cores. Deze experimentele chip, aangekondigd op 2 december 2009, zal vanaf zomer 2010 in beperkte aantallen geleverd worden aan onderzoeksinstellingen. De energieconsumptie is 25 tot 125 Watt, afhankelijk van het gebruik. Elk paar cores van de chip heeft een eigen, variabele klokfrequentie. Bron: http://techresearch.intel.com

de klok letterlijk terug te draaien: de hoge kloksnelheid van de chip moet weer naar beneden. De 4,7 Gigahertz van Huygens is wel het maximum bereikbare, met zo'n 5 miljard rekenoperaties per seconde. Een groter aantal operaties moeten we niet halen uit de klok, maar uit de multicore architectuur: meer rekenkernen, maar elk wat langzamer, zie Figuur 10. Dit vereist meer parallellisme, en dat komt goed uit, want daar hebben we de afgelopen jaren al veel ervaring mee opgedaan: als intellectuele uitdaging aan het front van de supercomputing, en in onze collegezalen waar we dit de afgelopen decennia al hebben onderwezen. Wie weet zorgt parallellisme als technologie ervoor dat we computerchips met een te hoog wattage kunnen verbieden, net zoals de LED-lamp en de spaarlamp het einde betekenden voor de veelverbruikende gloeilamp.

Snel

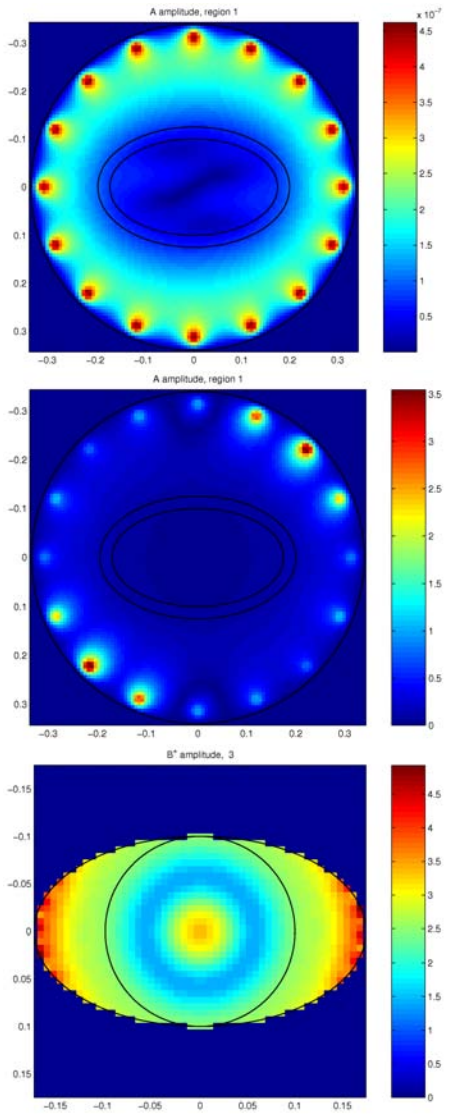
In februari 2007 was het departement wiskunde van de Universiteit Utrecht gastheer van de Studieweek Wiskunde en Industrie (SWI), een jaarlijks evenement waarin zo'n 60 wiskundigen van junior tot vergevorderde senior, van toegepast wiskundige tot zuiver wiskundige, zich werpen op een zestal problemen uit de omringende maatschappij. Samen met collega's Paul Zegeling, Karma Dajani, Tammo Jan Dijkema en Johan van de Leur heb ik deze week georganiseerd. Eén van de problemen werd aangedragen door het Universitair Medisch Centrum (UMC) in Utrecht. Dit academisch ziekenhuis had een nieuwe MRI-scanner aangeschaft met een magnetische veldsterkte van 7 Tesla, ruim meer dan de 3 Tesla van de vorige generatie. Daarmee kunnen betere beelden gemaakt worden van het menselijk lichaam, en bijvoorbeeld tumoren beter opgespoord worden. De scanner heeft 16 antennes die elk afzonderlijk afgesteld kunnen worden, zowel voor de fase als voor de amplitude, zie Figuur 11(a)–(b). Maar hoe moet dat? Het elektromagnetisch veld moet wel gelijk blijven op alle plaatsen in het onderzochte gebied van het lichaam.

De onderzoeksgroep van Nico van den Berg in het UMC had numerieke software om de afstelling te optimaliseren voor elke specifieke patiënt en om tegelijk binnen de strenge veiligheidsregels te blijven die plaatselijke opwarming van de patiënt moeten voorkomen. De numerieke methode heet *Finite-Difference Time-Domain* (FDTD) en leverde een behoorlijke verbetering op door de personalisering van de antenneafstelling. De bestaande methode was echter toch onbruikbaar in de praktijk omdat het zo'n 5 uur rekentijd op een pc kostte. Artsen hebben meestal niet zo veel geduld, en patiënten trouwens ook niet. Dat moest veel sneller. Een groep wiskundigen ging aan de slag en bedacht een snellere benaderde oplosmethode met behulp van Besselfuncties, goed genoeg qua optimalisering, maar met een oplossing binnen een halve minuut, zie Figuur 11(c). Dit werk is beschreven in de SWI-proceedings [18] en een populaire versie [12]. De studieweek had nog een nasleep: er verscheen een gezamenlijk wetenschappelijk artikel [19] in het tijdschrift *Physics in Medicine and Biology*, en uit de vervolgcontacten ontstond het afstudeeronderzoek van Alessandro Sbrizzi van de master Scientific Computing in Utrecht, begeleid door Gerard Sleijpen. Alessandro ontving in december 2009 zijn bul en is inmiddels begonnen aan een promotieonderzoek in het UMC, om daar zijn numerieke kennis in te zetten voor verdere verbeteringen en wie weet ook verdere versnellingen.

De maatschappij vraagt om wiskundige bijdragen. Zij heeft ons hard nodig. Tegelijk is ons werk niet altijd even zichtbaar. Een druk op de knop, de numerieke software doet zijn werk, de inwendige opwarming blijft minimaal en er komt een scherp plaatje uit de scanner. Bedenk bij het bekijken van dit soort plaatjes hoe groot de bijdrage van de scientific computing hieraan is geweest!

Ik wil u nog een ander voorbeeld geven waarbij snelheid een cruciale rol speelt, namelijk *matching*, het vinden van een geschikte partner voor alle leden van een groep. Denk aan de Joodse gemeenschap in het dorp Anatevka waar Hodel, de dochter van melkman Tevje, aan matchmaker Yente vraagt om in haar boek een perfecte match te vinden. De wensenlijst: allereerst moet het een *scholar* zijn, een geleerde. Maar liefst ook nog eens rijk.

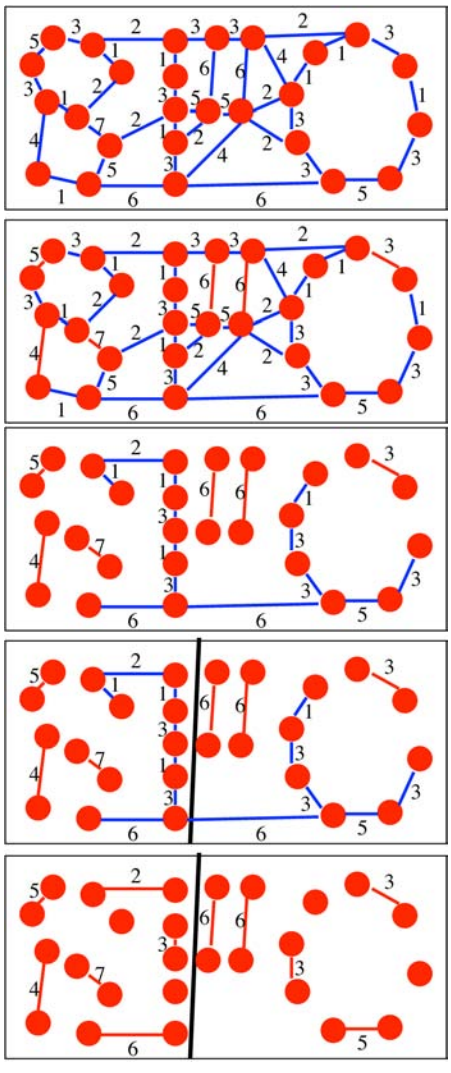
Laten we als wiskundigen even de rol van koppelaar spelen. In ons boek staan alle beschikbare kandidaten met hun gegevens en wensen. Als eerste stap kunnen we voor iedere kandidaat de lijst van mogelijke partners maken. Aantrekking die niet wederzijds is, gooien we uit de lijst. We houden zo wiskundig gezien een *graaf* over, een netwerk, met *knopen* (de kandidaten), en met *takken* ofwel verbindingen die de wederzijdse aantrekking representeren. Nu gaan we deze graaf wegen: op grond van de wensen en gegevens van elk mogelijk paar bepalen we een gewicht, de score die uitdrukt hoe goed de match is. Hoe hoger het gewicht, hoe beter de partners bij elkaar passen.



Figuur 11 Optimalisatie van het elektromagnetisch veld in de 7 Tesla MRI-scanner van het Universitair Medisch Centrum Utrecht [18].
(a) 16 antennes rond een elliptisch model van een buik;
(b) afzonderlijk afgestelde antennes;
(c) optimaal uniform geïnduceerd veld B^+: weinig warmteontwikkeling en minder artefacten in de MRI-beelden

Deze gewogen graaf, met n knopen en m verbindingen is de invoer van onze berekening. De graaf is ijl, want kandidaten hebben ieder een beperkt aantal mogelijke partners. Er moet immers liefde kunnen ontvlammen! Er zijn misschien $m = 10n$ verbindingen, maar zeker niet $m = n^2/2$, het maximaal mogelijke aantal. Als $n = 1000$ maakt dit al een verschil van 10.000 te verwachten versus 500.000 maximaal mogelijke verbindingen, wat dus een factor 50 scheelt. Voor een voorbeeld van een mogelijke graaf, zie Figuur 12(a).

Wat is een optimale oplossing voor Yente? Zoveel mogelijk personen aan de man of vrouw proberen te brengen? Dat is een simpeler probleem, waarbij de weegfactoren genegeerd worden, en het gebruik van de op-



Figuur 12 Ongerichte gewogen graaf met $n = 26$ knopen en $m = 38$ verbindingen. Elke verbinding heeft een gewicht dat de wederzijdse aantrekking representeert. Het totale gewicht is 120.
(a) De originele graaf;
(b) vijf (rode) verbindingen zijn dominant;
(c) gedomineerde verbindingen (buren van gekoppelde knopen) zijn verdwenen;
(d) een parallel algoritme voor twee processoren [11] veroorzaakt communicatie over de grens;
(e) een matching met totaal gewicht 50 berekend door het parallelle algoritme.

lossing daarvan belooft niet veel goeds voor de toekomst. De gewichten moeten echt meegenomen worden in de berekening. We willen een matching met het maximale totale gewicht. Dus de som van alle gewichten voor de paartjes die gekoppeld worden moet maximaal zijn. Dit probleem is in 1990 opgelost door Gabow [4], en kost aan rekentijd $O(mn+n^2 \log n)$. In die rekentijd krijg je een oplossing die aantoonbaar de beste is. We willen echter problemen oplossen met miljoenen knopen. Voor $n = 10^6$, en $m = 10^7$, is deze rekentijd ongeveer 3×10^{13} operaties. Op een moderne dual-core pc die 1 miljard operaties per seconde uitvoert op iedere core, kost dit 15.000 seconden rekenen, dus 4 uur en 10 minuten: veel te lang. Hier is grotere snelheid geboden. Dit ondanks de polynomiale tijd waarin de oplossing gevonden kan worden, het kost immers nooit meer dan $O(n^3)$. Er zijn moeilijker problemen zoals het handelsreizigersprobleem met n steden die bezocht moeten worden, en waarbij het aantal mogelijke volgordes gelijk is aan $n! = 1 \times 2 \times 3 \times \cdots \times n$. Dat groeit als $O(n!)$, dus veel harder.

In de praktijk stellen we ons tevreden met een oplossing die goed is maar niet optimaal. Algoritmen die een bepaalde kwaliteit garanderen heten *approximatie-algoritmen* of *benaderingsalgoritmen*. Robert Preis uit Paderborn bedacht in 1999 een algoritme dat gegarandeerd minstens de helft van het optimale gewicht oplevert in lineaire rekentijd, dus in $O(n)$ operaties [14]. In 2004 verbeterden Drake en Hougardy [3] en daarna Pettie en Sanders [13] dit naar een garantie van tweederde van het optimale gewicht, dus meer goede matches in dezelfde tijd, en wellicht betere huwelijken gearrangeerd door Yente.

Samen met Fredrik Manne van de universiteit van Bergen in Noorwegen heb ik gewerkt aan een parallel benaderingsalgoritme voor matching [11]. Het resultaat is gebaseerd op dominante verbindingen. Als jij mij het aantrekkelijkst vindt, en ik jou, dan kunnen wij het grote boek van Yente vergeten en hebben we een match, zie Figuur 12(b). De anderen hebben het nakijken en moeten verder in hun zoektocht, zie Figuur 12(c) Om dit herhaaldelijk uit te voeren hoeven we alleen naar onze mogelijke partners te kijken, en niet verder. Deze eigenschap bevordert parallellisatie. We doen dit op een zogenaamd bulk-synchrone manier. Iedere processor van onze computer heeft de verantwoordelijkheid voor n/p kandidaten, waarbij p het aantal processoren is. Als $p = 2$, dan is dat de helft van de kandidaten, zie Figuur 12(d). Iedere processor zoekt lokaal naar dominante verbindingen, koppelt paren, en wisselt daarna informatie uit met andere processoren. Dit betreft de afwijzingen ('niet meer beschikbaar'). Het proces herhaalt zich tot er niets meer te koppelen valt. Het eindresultaat is te zien in Figuur 12(e).

U ziet, wij gebruiken hierbij terminologie en methoden uit de combinatoriek en in het bijzonder de grafentheorie. Er is ook veelvuldige interactie met de theoretische informatica, een vakgebied dat voor ons van groot belang is. In de combinatorische scientific computing is er echter ook altijd een toepassingsaspect. Hier willen we de matchingsmethode toepassen, maar misschien niet direct voor Yente, want ze deed al goed werk. In onze moderne tijd zou u in de romantische sfeer blijvend aan online dating-services kunnen denken, maar ook aan medische toepassingen zoals het matchen van donoren en patiënten, of aan terreurbestrijding via het zoeken naar relevante verbindingen in grote sociale netwerken. In de Verenigde Staten is het Department of Homeland Security een drijvende kracht achter onderzoek naar netwerken. Als wetenschappers kunnen we hieraan bijdragen, maar zullen we ons ook bewust moeten zijn van de privacy-aspecten.

Een totaal ander vakgebied, de epidemiologie was recent veelvuldig in het nieuws door de nieuwe influenza H1N1, beter bekend als de Mexicaanse griep. De auteur is gespaard gebleven, behoorde niet tot een risicogroep, en is daarom niet gevaccineerd. Zo'n beslissing zou gebaseerd kunnen worden op netwerksimulaties die laten zien hoe je het beste bepaalde mensen (knopen in het netwerk) kunt vaccineren om de hele bevolking te beschermen, in beperkte tijd en met een beperkt aantal beschikbare vaccins. Hier ligt nog een mooi toepassingsterrein voor de scientific computing.

Doen wij in onze onderzoeksgroep op korte termijn nog iets met onze matchingsmethoden? Promovendus Albert-Jan Yzelman ontwikkelt dergelijke methoden met het oog op toepassing binnen Mondriaan. Het blijkt dat het grootste deel van de rekentijd in Mondriaan opgaat aan het kleiner maken van de matrix, voordat we deze splitsen. We matchen kolommen die nogal op elkaar lijken, vanwege hun niet-nullen in dezelfde rijen, of vrijwel in dezelfde rijen. Als dat sneller en beter kan, dan kan Mondriaan ook sneller en beter worden. En dan kunnen we parallelle programma's zoals het osteoporose-programma uit Zürich op hun beurt weer sneller maken!

Toegepaste wiskunde

Buiten de academische wereld heb ik contacten met bedrijven zoals Keygene, NXP, Ortec, Philips en VORtech, in het kader van stagebegeleiding, advieswerk, of onderzoekssamenwerking. Bij Shell heb ik in het verleden zes jaar gewerkt. Toepassingen in bedrijven zijn belangrijk voor ons. Het geeft een goed gevoel als je niet alleen een wiskundige methode bedenkt, maar ook ziet dat die direct van nut is. De toepassingen helpen onze studenten interessante banen te vinden. En de studenten helpen die bedrijven weer vooruit via hun analytisch vermogen.

Rond de financiering van onderzoek waait een gure wind. De overheid wil wel geld spen-

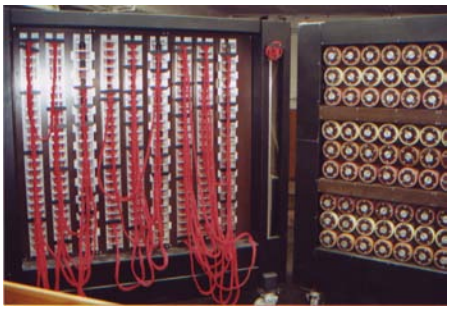
deren aan het hoger onderwijs en het universitaire onderzoek, maar ziet een groeiend begrotingstekort. Het bedrijfsleven merkt de gevolgen van de kredietcrisis, maar zal er na een tijdelijk dipje vast weer bovenop komen. Ik roep het bedrijfsleven op om de universiteiten meer te steunen. Wees niet zuinig, stel beurzen in voor masterstudenten, betaal eens mee aan een promovendus. Bent u tevreden over al die afgestudeerden in de scientific computing die bij u werken? Stel beurzen voor getalenteerde studenten in om die opleiding te versterken. En dat geldt natuurlijk ook voor andere masteropleidingen, hier en elders in het land.

De toegepast wiskundige wordt gedreven door dezelfde nieuwsgierigheid als beoefenaars van de zuivere wiskunde. Toch wordt deze nieuwsgierigheid in de zuivere wiskunde vaak anders beleefd. Godfrey Harold Hardy, de grote Britse getaltheoreticus uit het Cambridge van de vroege twintigste eeuw, die niets moest hebben van toepassingen van de wiskunde, en ze nogal saai en elementair vond, schreef in 1940 in *A Mathematician's Apology* [5] onder andere

"I have never done anything 'useful'. No discovery of mine has made, or is likely to make, directly or indirectly, for good or ill, the least difference to the amenity of the world."

Vrij vertaald: "Ik heb nooit iets 'nuttigs' gedaan. Geen enkele van mijn uitvindingen heeft enig verschil uitgemaakt of zal dat doen, direct of indirect, in positieve of negatieve zin, voor het gemak van de wereld"

Dat was aan het begin van de Tweede Wereldoorlog die enkele jaren later mede door wiskundigen zoals Alan Turing werd gewonnen. Zijn toegepast onderzoek was het helpen ontwerpen van een machine, de Colossus, de voorloper van de moderne computer, zie Figuur 13, met als doel de Duitse geheime codes te breken. Laten we de geest van Hardy, die hier en daar nog waart, verjagen en laten we vooral de geest van Turing volgen,



Figuur 13 Replica van de Colossus, voorloper van de moderne computer, 1943–44, ontworpen door Alan Turing c.s. in Bletchley Park, VK. Zie [7] voor een biografie van Turing.

door spannende wiskunde te bedrijven, met een oog voor mogelijke toepassingen, soms met behulp van de computer. Laten we vooral niet proberen onszelf te verkopen als kunstenaar, als *dandy* of als *man of leisure* op afstand van de werkelijke wereld, maar juist als wiskundigen die volop in de moderne maatschappij staan, hun steentje bijdragen aan de leniging van menselijke noden waar dat kan, en tegelijk een prachtig vak beoefenen.

Referenties

1 C. Bekas, A. Curioni, P. Arbenz, C. Flaig, G.H. van Lenthe, R. Müller, and A.J. Wirth, 'Extreme scalability challenges in micro-finite element simulations of human bone', *Proceedings International Supercomputing Conference (ISC 2008), Dresden, Germany* (2008).

2 R.H. Bisseling, *Parallel Scientific Computation: A Structured Approach using BSP and MPI*, Oxford University Press, Oxford, UK, 2004.

3 D.E. Drake and S. Hougardy, 'A linear-time approximation algorithm for weighted matchings in graphs', *ACM Transactions on Algorithms*, **1** (2005), pp. 107–122.

4 H.N. Gabow, 'Data structures for weighted matching and nearest common ancestors with linking', *Proceedings First Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 434–443.

5 G.H. Hardy, *A Mathematician's Apology*. Cambridge University Press, Cambridge, UK, 1940.

6 B. Hendrickson and A. Pothen, 'Combinatorial scientific computing: The enabling power of discrete algorithms in computational science', *VECPAR 2006*, *Lecture Notes in Computer Science* **4395** (2007), pp. 260–280. Springer-Verlag, Berlin.

7 A. Hodges, *Alan Turing: The Enigma*, Burnett Books, London, UK, 1983.

8 G. Karypis, V. Kumar, 'A parallel algorithm for multilevel graph partitioning and sparse matrix ordering', *Journal of Parallel and Distributed Computing*, **48**(1) (1998), pp. 71–95.

9 B.W. Kernighan, S. Lin, 'An efficient heuristic procedure for partitioning graphs', *Bell System Tech. J.*, **49** (1970), pp. 291–307.

10 A.N. Langville, C.D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, Princeton, NJ, 2006.

11 F. Manne, R.H. Bisseling, 'A parallel approximation algorithm for the weighted maximum matching problem', *Proceedings Seventh International Conference on Parallel Processing and Applied Mathematics (PPAM 2007)*, *Lecture Notes in Computer Science* **4967** (2008), pages 708–717.
De getoonde graaf werd voor het eerst gepresenteerd op de SIAM Conference on Parallel Processing for Scientific Computing, 25–27 februari 2004, San Francisco, USA.

12 B. Mols, 'Minder verstoringen in MRI-beelden', *Nieuw Archief voor Wiskunde*, **9**(1) (2008), pp. 58–59.

13 S. Pettie, P. Sanders, 'A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching', *Information Processing Letters*, **91** (2004), pp. 271–276.

14 R. Preis, 'Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs', *Proceedings STACS '99*, *Lecture Notes in Computer Science* **1563** (1999), pp. 259–269. Springer-Verlag, Berlin.

15 Een lagere kloksnelheid betekent dus *veel minder* elektriciteitsverbruik. Het dynamische deel van het verbruik schaalt als $O(CV^2f)$ [16], met C de capaciteit, V het voltage en f de frequentie (kloksnelheid). Verlaging van de frequentie maakt het mogelijk ook het voltage te verlagen en zo meer dan proportioneel te besparen in het verbruik. We hebben dus liever 2 cores met de halve kloksnelheid dan 1 core met de hele kloksnelheid.

16 J.M. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits*, Prentice Hall, US, second edition, 2003.

17 M.A. Stijnman, R.H. Bisseling, G.T. Barkema, 'Partitioning 3D space for parallel many-particle simulations', *Computer Physics Communications*, **149**(3) (2003), pp. 121–134.

18 J.B. van den Berg, N. van den Berg, B. van den Bergen, A. Boer, F. van de Bult, S. Dahmen, K. Frederix, Y. van Gennip, J. Hulshof, H. Meijer, P. in 't Panhuis, C. Stolk, R. Swierstra, M. Veneroni, E. Vondenhoff, 'Understanding the electromagnetic field in an MRI scanner', *Proceedings 58th European Study Group Mathematics with Industry Utrecht 2007*, pp. 69–90. Universiteit Utrecht, 2007, editors: R.H. Bisseling, K. Dajani, T.J. Dijkema, J. van de Leur, P.A. Zegeling.

19 B. van den Bergen, C.C. Stolk, J.B. van den Berg, J.J. Lagendijk, and C.A. van den Berg, 'Ultra fast electromagnetic field computations for RF multi-transmit techniques in high field MRI', *Physics in Medicine and Biology*, **54**(5) (2009), pp. 1253–1264.

20 A. van Heukelum, G.T. Barkema, R.H. Bisseling, 'DNA electrophoresis studied with the cage model', *Journal of Computational Physics*, **180**(1) (2002), pp. 313–326. De cage matrices zijn verkrijgbaar via de University of Florida Sparse Matrix Collection van Tim Davis, www.cise.ufl.edu/research/sparse/matrices/vanHeukelum .

21 B. Vastenhouw, R.H. Bisseling, 'A two-dimensional data distribution method for parallel sparse matrix-vector multiplication', *SIAM Review*, **47**(1) (2005), pp. 67–95. De Mondriaan software is vrij te verkrijgen via www.math.uu.nl/people/bisseling/Mondriaan.

22 A.N. Yzelman, R.H. Bisseling, 'Cache-oblivious sparse matrix–vector multiplication by using sparse matrix partitioning methods', *SIAM Journal on Scientific Computing*, **31**(4) (2009), pp. 3128–3154. Matrixpermutaties en matrixvisualisaties zullen onderdeel uitmaken van Mondriaan versie 3.0, verwacht in het voorjaar van 2010.