EVOLUTIE

SO, WE ALL EVOLVED FROM TOASTERS?
YES, UNLESS YOU BELIEVE IN SCIENTISTS AND ENGINEERS...
BRAINSTORM

INHOUDSOPGAVE

Colofon

De Brainstorm is een uitgave van Studievereniging Cover en wordt verspreid onder leden, staf en anderszins geïnteresseerden. De Brainstorm verschijnt minimaal 3 keer per jaar, in een oplage van 500 stuks.

Redactieadres
Studievereniging Cover
t.a.v. De Brainstorm
Postbus 407
9700 AK Groningen
brainstorm@svcover.nl
www.svcover.nl

Redactie
Voorzitter — Eric Jansen
Eindredacteur — Maarten van Gijssel
Secretaris — Aliene van der Veen
Penningmeester — Eveline Broers
Algemeen lid — Sjors Lutjeboer

Lay-out — Maarten van Gijssel, Eric Jansen

Adverteerders — Studystore, Café Karakter, Sogeti

VAN DE REDACTIE

AUTEUR: *Eric Jansen, redactielid*

Toen ik aan boord was van de H.M.S. Brainstorm, als redactionist, werd ik gegrepen door bepaalde feiten over de verschillen tussen de vorige edities en over de cosmetische relatie van latere edities ten opzichte van eerdere. Deze feiten leken in mijn ogen licht te werpen op de oorsprong der Brainstorms - dat mysterie der mysteries.

Alles evolueert

Er was een tijd dat de Brainstorm een zwart-wit blaadje was met heel veel fluff (onze term voor inhoud die niet studiegerelateerd is). De laatste paar jaar is de Brainstorm echter behoorlijk geëvolueerd. We zijn naar een lay-out gegaan met twee kolommen in plaats van één, we zijn steeds meer op zoek gegaan naar studiegerelateerde content en we zijn van zwart-wit via een paar pagina's in kleur uiteindelijk bij een full-color blaadje gekomen. Nu zullen de meesten van jullie alleen Brainstorms hebben gekend die allemaal vrijwel exact dezelfde lay-out hadden. Inmiddels drukken we al een tijdje in kleur en leek het ons leuk om te proberen hier iets spannenders mee te doen. Dat heeft tot gevolg gehad dat de Brainstorm die je nu leest een nieuwe en frisse uitstraling heeft. Dat betekent niet dat het een totaal ander blaadje is geworden; je kan nog steeds genieten van verscheidene inhoudelijke en minder inhoudelijke rubrieken die ons blaadje rijk is.

Dat we ook altijd bezig zijn met het evalueren en laten evolueren van onze inhoud is bijvoorbeeld terug te zien in de nieuwe rubriek De switch, die de afgelopen editie is geïntroduceerd. In deze editie wisselen twee Coverleden een film uit waar ze zelf iets mee hebben. Daarnaast is dit de laatste keer dat we een strip plaatsen van vaste striptekenaar Andries de Vries. Helaas stopt hij na deze editie met het maken van strips voor de Brainstorm.

Gelukkig kun je wel blijven genieten van een aantal andere vaste rubrieken. Naar aanleiding van het thema schrijft KI-docent Marco Wiering een stukje over evolutionaire algoritmes, vaste columnist Jaap neemt de ware aard van de evolutie onder de loep en Informatica-alumnus Pjotr Svetachov vertelt iets over de huidige evolutie van programmeerstijlen.

Tot slot besteden we ook wat aandacht aan de evolutie van onze vereniging (vroeger zat Informatica bijvoorbeeld helemaal niet bij Cover!) en hebben we een grafisch overzicht van de evolutie van de Brainstorm door de jaren heen. Geniet van deze vernieuwde editie van de Brainstorm, misschien ondanks, maar hopelijk dankzij, het nieuwe jasje!

VAN HET BESTUUR

AUTEUR: *Maikel Grobbe, secretaris Cover*

Evolutie, de theorie die Darwin bedacht en later in zijn boek "On the origin of species by means of natural selection" uitgebreid beschreef. Ikzelf geloof wel in deze evolutietheorie en ben erg dankbaar dat de natuur op deze manier werkt. Wij hebben ons immers door de samenloop van omstandigheden geëvolueerd tot de soort die wij heden ten dage zijn.

Zonder evolutie waren we nu nog een bacterie geweest die zich ergens in de oceaan bevond. Zonder duimen, oren, ogen, benen, armen en allerlei andere handige dingen. Waarvoor ik toch echter wel het meest dankbaar ben zijn onze hersenen. Onze hersenen waarmee wij allerlei complexe problemen oplossen en tot geniale uitvindingen komen. De Brainstorm, een raket waarmee we naar de maan kunnen afreizen en natuurlijk niet te vergeten: Pokémon. Pokémon, een geniaal concept, waar evolutie ook een grote rol speelt. Zonder evolutie waren we immers nergens geweest en hadden we nooit kennis gemaakt met Venusaur, Charizard en Blastoise, maar hadden we voor altijd Bulbasaur, Charmander en Squirtle gehouden. Niemand die daar blij van was geworden, toch?

Ook wij als bestuur hebben met evolutie te maken en je zou zelfs kunnen zeggen dat wij gedurende het bestuursjaar evolueren. Een bestuur begint als een zielige en weerloze Caterpie, die iedereen nog erg schattig vindt. Iedereen past nog op met wat ze doen, zodat het nieuwe bestuur niet wordt gekwetst. Op een gegeven moment is het echter klaar met deze zielige kwetsbare periode en is het tijd dat het bestuur evolueert. De leden krijgen namelijk door dat sommige dingen niet zo goed gaan als dat ze zouden kunnen gaan en beginnen met het geven van kritiek. De evolutionaire stap die het bestuur dan maakt is die naar de Metapod-fase, waarin zij zich wapent tegen deze kritiek door een schild om zich heen te bouwen dat een deel van de kritiek absorbeert. Dit stadium is echter weinig productief en het is dan ook taak voor een bestuur om zich zo snel mogelijk te ontpoppen tot de Butterfree-fase waarin ze iedereen verblindt met de pracht en praal die ieder bestuur in zich heeft. In het Butterfree-stadium van elk bestuur weet iedereen precies wat er moet gebeuren en gaat alles veel makkelijker dan in eerdere stadia. Het bestuur is volgroeid en krachtiger dan ooit en kan zich gaan richten op het voortbrengen van nieuwe Caterpie die zich kunnen gaan ontwikkelen tot een mooie Butterfree.

Pokémon, een geniaal concept

AUTEUR: Pjotr Svetachov, alumnus Informatica

Elke editie vragen wij een alumnus wat te vertellen over hun afstudeeronderzoek, het leven na het studeren of iets wat hen bezig houdt. Deze keer Pjotr Svetachov over DOD: Data Oriënted Design.

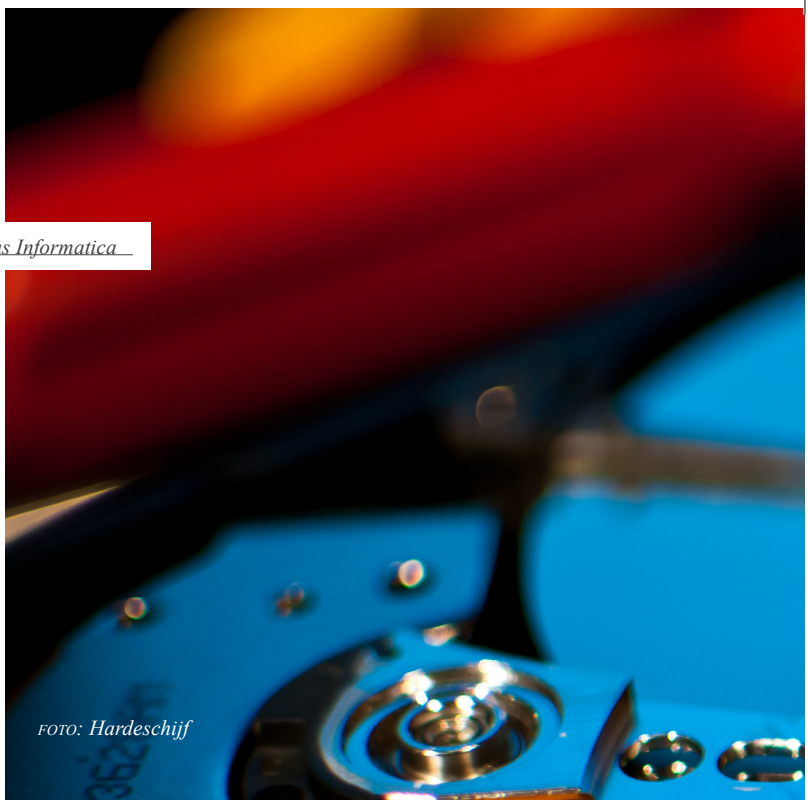


FOTO: Hardeschijf

Mensen houden niet van trage software en daarom word je als informaticus al snel in je opleiding doodgegooid met verschillende algoritmen. Alleen voor het sorteren heb je al bubble sort, insertion sort, merge sort, quick sort en radix sort. Ook worden je verschillende programmeerstijlen aangeleerd in vakken zoals imperatief, functioneel en objectgeorinteerd programmeren. Dit allemaal moet je een soort van mindset geven. In dit rijtje mist een programmeerstijl die in de laatste drie jaar heel populair is geworden, vooral bij het programmeren van computerspellen: Data Oriented Design.

Data Oriented Design (DOD) is een heel uitgebreide stijl van programmeren. Omdat het nog zo nieuw is, is nog niet vastgesteld wat er wel en wat er niet bijhoort. Globaal gezien maak je designkeuzes bij DOD vanuit de architectuur waarvoor geprogrammeerd wordt en de data die bewerkt moet worden. Ik zal in dit artikel eerst uitleggen hoe DOD ontstaan is. Daarna ga ik aan de hand van een paar voorbeelden proberen uit te leggen wat het precies inhoudt.

Stukje geschiedenis

Zoals eerder gezegd staat de architectuur heel erg centraal bij DOD. Dit komt door de evolutie die computersystemen gemaakt hebben. Iedereen kent wel de wet van Moore. Deze zegt dat elke 18 maanden het aantal transistors op een chip verdubbelt. Meestal wordt hier meteen bij gezegd dat de snelheid van computers dus ook moet verdubbelen. Dit is helaas niet waar. Kijk bijvoorbeeld naar de CPU. Het aantal cores dat op een

Data Oriented Design

CPU zit gaat steeds omhoog, maar de snelheid van een enkele core is al een tijdje bijna hetzelfde gebleven. Hetzelfde is ook aan de hand met het geheugen, alleen een stapje erger: de snelheid van het geheugen lijkt al lang geleden zijn maximum te hebben bereikt. In principe betekent dit dat de data-overdracht van het geheugen naar de processor heel traag is. Gelukkig is hier wat op bedacht: de cache. Er zitten meerdere lagen van caches tussen het geheugen en de CPU. Caches zijn in elke laag steeds kleiner en sneller. Je CPU wordt heel erg blij als de data die nodig is al in de cache zit, want anders moet hij wachten op de data. De huidige generatie CPU's heeft hier wat op gevonden: out-of-order execution. Als de CPU ergens op wacht kijkt hij of hij alvast een andere instructie kan uitvoeren die onafhankelijk is van de huidige instructie.

Nu heb ik eerder gezegd dat DOD vooral in de spellenindustrie wordt toegepast. Dit heeft een reden. Bij DOD hangt je design vooral af van de architectuur en in de spellenindustrie spelen consoles een heel belangrijke rol. De Xbox 360 is bijvoorbeeld al bijna zeven jaar oud. Ontwikkelaars hebben dus al zeven jaar de tijd gehad om de architectuur helemaal te begrijpen en hun code ervoor te optimaliseren. Je kan het ook anders zien: de ontwikkelaars programmeren op een zeven jaar oude CPU en deze mist belangrijke functies, onder andere out-of-order execution. Om toch het uiterste eruit te halen, waren de programmeurs gedwongen om hun algoritmen zo aan te passen dat de CPU zo min mogelijk hoeft te wachten.

Let op je data

Om de werking van een cache te laten zien zal ik een voorbeeld geven. Het komt voor dat je data hebt die eigenlijk een 2D grid moet voorstellen. Goede voorbeelden hiervan zijn afbeeldingen: elk datapunt is een pixel. Je kan ook met 3D grids te maken hebben als je bijvoorbeeld volumedata, zoals MRI-scans, bewerkt. Voor dit voorbeeld houden we het bij 2D. In het geheugen hebben we een array data gedeclareerd. Deze moet een grid voorstellen van width*height. Omdat een array 1D is en we er 2D data in willen stoppen, coderen we de data zo dat element i,j op plek i+j*height staat. Stel, je doet nu een simpele berekening op de data die afhangt van i en j zoals data[i+j*height] += i*j. Veel c/c++-code die ik tegenkom ziet er ongeveer zo uit:

```
for(int i = 0; i < width; i++){
        for(int j = 0; j < height;j++){
                data[i+j*height] += i*j;
        }
}
```

Nu heb ik het bovenste voorbeeld op mijn computer gedraaid met width = 4096 en height = 4096. Gemiddeld doet mijn computer er 0.255 seconden over. Nu heb ik deze code omgeschreven tot:

```
for(int j = 0; j < height;j++){
        for(int i = 0; i < width; i++){
                data[i+j*height] += i*j;
        }
}
```

Als ik deze code uitvoer dan doet mijn computer er gemiddeld 0.018 seconden over. Dat is ongeveer 14 keer zo snel! En dit terwijl de code boven en onder dezelfde complexiteit heeft. Hoe dit komt heeft te maken met hoe de cache van een processor werkt.

De cache in een processor bestaat uit meerdere cache lines. Elke line heeft een bepaalde grootte (bijvoorbeeld 64 bytes). Als de processor data uit het geheugen opvraagt en deze zit niet in de cache (ook wel cache miss genoemd), dan wordt dat stukje data en de data eromheen in één van de cache lines van de CPU gezet. Als je nu naar het voorbeeld kijkt, dan valt meteen op dat in het eerste stukje code de CPU steeds in stapjes van height door je data gaat terwijl in het tweede voorbeeld het steeds stapjes van één zijn. In het tweede voorbeeld heb je dus veel meer kans dat de data die je opvraagt al in de cache aanwezig is (cache hit). Als ik nu width = 4096*8 en height = 4096/8

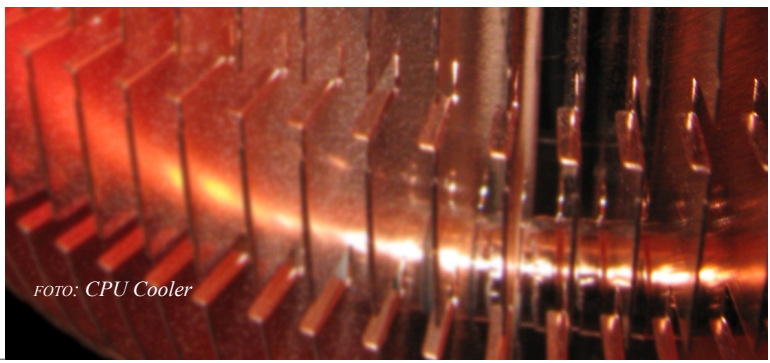


FOTO: CPU Cooler

neem, dan doet mijn eerste voorbeeld er ineens 0.09 seconden over. Waarschijnlijk omdat de CPU nu vaker een cache hit heeft.

In het bovenstaande voorbeeld heb ik het algoritme aangepast om meer cache hits te krijgen. Ik had het ook anders kunnen doen, namelijk door de originele data aan te passen. Had ik bijvoorbeeld van te voren bedacht dat het punt i,j op plek j+i*height staat, dan had het eerste algoritme meer cache hits dan de tweede. Bij DOD denk je dus niet alleen aan je algoritmen, je moet ook grote beslissingen nemen over hoe je je data opslaat. Bomen en lijsten zijn vaak heel leuke structuren om je data in op te slaan, maar deze bestaan meestal uit allemaal pointers door je geheugen heen en dat is slecht voor de cache.

Van sommige talen is het ook niet meteen duidelijk hoe de code die jij tikt omgezet wordt naar machinecode

Naast dat je weet hoe de CPU werkt, moet je ook weten hoe je compiler werkt. Als de compiler slim genoeg was, had hij immers de bovenste code al voor mij kunnen optimaliseren. Van sommige talen is het ook niet meteen duidelijk hoe de code die jij tikt omgezet wordt naar machinecode. Denk hierbij maar aan functionele talen zoals Haskell of wat hogere talen zoals mathematica en matlab.

Structure of arrays

In het bovenste voorbeeld hadden we een triviale aanpassing. Je kan hier echter heel erg ver in gaan. Ik kom nu met een voorbeeld dat waarschijnlijk veel harten van hardcore object-oriented programmeurs stil zet (nee echt, ik raad ze aan om de volgende twee paragrafen over te slaan). Stel, je hebt een simulatie waar je objecten hebt met een positie, snelheid en twintig andere eigenschappen. Deze objecten kunnen van alles zijn: atomen, moleculen, personen of sterren; het maakt niet uit voor dit voorbeeld. Meestal maken programmeurs een struct/class die eruitziet zoals dit:

```
struct vec3{
        float x,y,z;
}

struct Object{
        vec3 position;
        vec3 velocity;
        //nog 20 eigenschappen
        ....
};

Object object[10000]; // 10000 objecten
```

In deze simulatie ga je elk tijdsinterval dt nieuwe posities berekenen aan de hand van de snelheid. Wat je dus in feite doet is steeds de positie en snelheid uitlezen en de andere twintig eigenschappen overslaan. Elke compiler daarentegen zal de data van de hele struct bijelkaar in het

geheugen stoppen. Deze twintig eigenschappen worden dus wel naar je cache verplaatst. Het zou misschien dus beter voor je cache zijn om de posities en de snelheden als een aparte array in het geheugen te hebben, los van het object. In plaats van een array van allemaal structures (Array of structures ofwel AoS) heb je dan nu een struct met allemaal arrays:

```
struct Objects{
        vec3 position[10000];
        vec3 velocity[10000];
        //nog 20 eigenschappen,
        //allemaal in arrays
        ...
}
Objects objects;
```

Dit werkt omdat de CPU meerdere cache lines heeft, de posities en velocities komen dus beiden in de cache terecht. Deze methode wordt ook wel Structure of Arrays (SoA) genoemd. Dit lijkt een beetje overdreven en niet meer objectgeoriënteerd, maar geloof het of niet: in drie jaar tijd wist deze programmeerstijl de hele spelindustrie te besmetten, je hoort nu bijna op elk congres wel een praatje waar men het woord SoA laat vallen. Natuurlijk maakt deze manier van programmeren het onderhouden van de code veel lastiger. Je moet dit soort ontwerpen dus alleen maar toepassen op plekken in je code waarvan je zeker weet dat het daar zin heeft. Dit kan je doen door een profiler te gebruiken (AMD CodeAnalyst is gratis en werkt ook op Intel processoren). Een profiler kan bijvoorbeeld laten zien hoe vaak functies worden aangeroepen of hoelang je programma over sommige regels code doet. Meestal zijn er meerdere redenen die je ertoe dwingen om je data in SoA-formaat op te slaan. Eén van de redenen kan bijvoorbeeld zijn het gebruik van SSE- of SSE2-instructies. Dit zijn speciale CPU-instructies die op meerdere getallen tegelijk werken, mits deze getallen naast elkaar in het geheugen staan.

In mijn laatste voorbeeld zag je wat een groot contrast er kan bestaan tussen DOD en OOP. Dit hoeft niet te betekenen dat DOD je code ongestructureerd maakt. Wat het wel betekent is dat je anders over je design moet nadenken. In plaats van dat je allemaal objecten hebt die zichzelf onderhouden, heb je nu allemaal componenten waar data in- en uitgaat. Je kan je programma dan meer gaan beschouwen als een lopende band.

Deze stijl evolueert dus ook met de architectuur mee. Als programmeur is het dus je taak om bij deze evolutie bij te blijven.

Ik heb nu een paar voorbeelden genoemd van DOD, maar zoals ik al aangaf is deze stijl van programmeren heel uitgebreid en niet in één artikel samen te vatten. De cache is namelijk niet het enige in de CPU waar je om moet denken. De PS3 heeft bijvoorbeeld de Cell processor met zijn eigen speciale instructies. Ook gaan er geruchten dat de volgende generatie consoles meer processoren krijgt voor speciale taken zoals physics. Deze stijl evolueert dus ook met de architectuur mee. Als programmeur is het dus je taak om bij deze evolutie bij te blijven.

FEESTJE?

WIJ DENKEN GRAAG MEE

Café Karakter is de ideale locatie voor:
- (Afstudeer) borrels of (verjaardags) feesten
- Activiteiten en/of borrels van je (studie) vereniging
- Bedrijfsfeesten en -borrels

KIJK VOOR MEER INFORMATIE OP: **WWW.CAFEKARAKTER.NL**

KLEINE PELSTERSTRAAT 6
GRONINGEN
T: 050-3187566

CAFE
KARAKTER
WWW.CAFEKARAKTER.NL

COLUMN JAAP

AUTEUR: *Jaap Oosterbroek*

Veel van mijn lezers hebben om een of andere reden het idee dat er zoiets als evolutie bestaat: het idee dat door een combinatie van selectie, druk en (semi-) willekeurige mutatie alles altijd maar beter aangepast wordt. De populariteit van dit belachelijke en bovenal onwaarschijnlijke idee is zo groot dat zelfs informatici en cognitiewetenschappers allemaal problemen proberen op te lossen met zogenaamde evolutionaire algoritmen. En alsof al deze beledigingen van ons intellect nog niet voldoende waren, kwam meneer Dawkins, opperprotagonist van de evolutie, laatst een populair wetenschappelijk praatje geven aan onze universiteit, nota bene binnen een decennium na de dood van zijn grote tegenstander: de eerbare heer Gould.*

Gezien meneer Gould momenteel helaas onbeschikbaar is voor een weerwoord, vrees ik dat de taak aan mij valt om jullie te verlossen van deze verleidelijke, maar o zo ongegronde verzameling misconcepties. Beste medestudenten, er bestaat GEEN evolutie, er is ENKEL degeneratie.

Degeneratie is het proces van het geleidelijk verval van ons eens perfecte genetische materiaal. Het is de kill hand van de entropie die tegelijkertijd in elke cel van ons lichaam grijpt en daar zijn dodelijke ioniserende dans doet met basenpaartjes en chromosoomverdelingen. Laat mij het uitleggen. Ooit was er een onbewogen beweger, daar kunnen we het allemaal over eens zijn. Laten we deze onbewogen beweger God** noemen. Sterrenkundigen menen dat God aan het begin van het universum de oerknal veroorzaakte*** Dit is echter een wonderlijk idee. Waarom zou God op dat punt van onze eventuele geschiedenis inspringen? Hoewel de aantrekkingskracht van een grote knal op infantiele geesten zoals sterrenkundigen goed te verklaren is, lijkt het uiteraard veel logischer dat een eventueel intelligente God inspringt op een spannender tijdstip, bijvoorbeeld 8000 jaar voor de geboorte van Christus****. In den beginne schiep God daar een briljant ecosysteem van mensen, dieren en andere volmaakte organismen. Als finishing touch voegt hij ook nog het geheel van fossielen, planten en achtergrondstraling toe*****. Na een paar uur spelen met deze creaturen raakt God verveeld en verlaat het geheel om in de eindtijd nog maar eens te komen kijken of er nog iets van over is.

Ooit was er een onbewogen beweger, daar kunnen we het allemaal over eens zijn.

Ondertussen vervallen de cellulaire machines die het geheel doen draaien in rap tempo. Evolutietheorie claimt dat de kans dat een beter aangepast organisme blijft

leven groter is dan dat een slecht aangepast organisme blijft leven. Dit is een zeer geloofwaardige aanname. Maar hoe groot is de kans dat een organisme beter aangepast raakt door een mutatie? Op eiwitniveau is de kans dat een mutatie een negatief gevolg heeft velen malen groter dan de kans dat deze een positief gevolg heeft. En welke kans is nu groter? Als we deze conditionele kansen meenemen, ontstaat er een ander beeld van willekeurige mutatie. In een eindige populatie met matig consistente selectiedruk is de kans dat er een positieve mutatie optreedt en deze overleeft vele malen kleiner dan de kans dat er een negatieve mutatie optreedt en deze overleeft. Zodoende lijden alle organismen aan degeneratie.

Voor sommigen is puur wiskundige theorie natuurlijk niet voldoende. Maar voor zij die durven te kijken is degeneratie natuurlijk altijd en overal zichtbaar. Walvissen die aanspoelen omdat ze niet meer snappen dat stranden gevaarlijk zijn, slechtziende vogels die tegen spiegelende gebouwen en windmolens aanvliegen, eerstejaars die ondanks verwoede pogingen nooit zullen leren wave-dashen. Helaas betekent degeneratie ook dat ons gedachtegoed tot een langzame teloorgang veroordeeld is. Voor sommigen, zoals meneer Dawkins, betekent degeneratie zodanige verminderde hersenfunctie dat ze de evidente foutheid van hun eigen ideeën niet meer kunnen inzien.

Voor anderen betekent degeneratie dat hun lichaam hen in de steek laat, zodat zij niet langer weerwoord kunnen bieden. In een laatste heroïsche poging zijn punt te bewijzen stierf meneer Gould op 20 mei 2002 aan kanker: de degeneratie van samenwerking binnen het lichaam. Geschiedenis wordt geschreven door zij die toevallig het laatst de dood vinden.

* Voor opmerkingen over lange onbegrijpelijke zinnen: zie eerder werk in deze serie columns.
** Mensen zonder dramatische aanleg mogen de hoofdletter op deze naam wegdenken als hij zich niet aan het begin van een zin bevindt.
*** Sommigen worden opstandig als je dat tegen ze zegt en slaan je dan om je oren met formules en moeilijke woorden die uiteindelijk op hetzelfde neer komen.
**** Al dan niet zijn gedegenereerde nazaat.
***** Ruw werk kost weinig moeite en ik durf te wedden dat als je al die sterren en nevels van dichtbij gaat bekijken het stiekem kartonnen platen zijn.

P.S.
Om opnieuw mores met mijn afstudeerbegeleider te vermijden, moet ik, vrees ik, expliciet benadrukken dat deze column een werk van ongenuanceerde fictie is en ik mij op geen enkele consequent voorspelbare wijze verbind aan of distantieer van standpunten in deze tekst.

FOTO: Fossiel

Evolutionary Algorithms

AUTEUR: Marco Wiering, docent Kunstmatige Intelligentie

Evolutie vind je niet alleen terug in de natuur, ook in KI speelt het een belangrijke rol. Marco Wiering vertelt over de theorie van 'Evolutionary Algorithms'.

Inspired by the success of nature in evolving such complex creatures as human beings, researchers in artificial intelligence have developed algorithms based on evolution theory. The class of these methods is called evolutionary computing and consists among others of genetic algorithms, evolutionary strategies, and genetic programming. Evolutionary algorithms are optimization algorithms that are inspired by Darwin's evolution theory, known as natural selection or survival of the fittest, and they were invented during the 1960s and 1970s. One of their strengths is that they can find very good solutions in very large search spaces, where exhaustive search (trying out all possible solutions) would cost way too much time.

In evolutionary algorithms a population of solutions is evaluated after which the best solutions are allowed to reproduce most offspring (children). This is then repeated for many generations. The idea is that if the parent individuals form good solutions, they are likely to possess good building blocks of genetic material (the genetic material makes up the solution) that may be useful for creating new individuals. Genetic algorithms usually take two parent individuals and they recombine their genetic material to produce a child. If the child performs well on the evaluation test, it will also be selected for reproduction and in this way the genetic material can again be propagated to new generations. Since in evolution the individuals themselves will die, Richard Dawkins came up with the selfish gene hypothesis in 1976, which says that basically the genes are alive and use the mortal individuals as hosts so that they are able to propagate themselves further. Some genes may be found in many individuals, whereas other genes are only found in a small subset of individuals. In this way, the genes compete for hosts, and genes which occupy well performing individuals are likely to reproduce themselves. We will now first describe optimization problems and then examine which steps should be pursued for constructing an evolutionary algorithm.

Solving optimization problems

A lot of research in computer science and artificial intelligence has been devoted to solving optimization problems. There are many different optimization problems: shortest path planning, for example, requires the algorithm to compute the shortest path from a state to a particular goal state. Well known applications are planners used by cars or for train passengers. In principle, shortest path problems are simple problems and can be solved efficiently by Dijkstra's shortest path algorithm or the A* algorithm. These algorithms can compute the shortest path in a short time for problems consisting of more than 1,000,000 cities (or nodes if we formalize the problem as a graph using nodes and weighted edges representing the distances of connections between nodes). On the other hand, there also exist combinatorial optimization problems and these are very hard to solve. One example is the traveling salesman problem (TSP). This problem requires that a salesman goes to N customers that live in different cities, so that the total tour he has to make from his starting city to all customers and back to his starting place has a minimal length.

This problem is known to be NP-complete and therefore unless P = NP not solvable in polynomial time. For example, if we use an exhaustive search algorithm which computes and evaluates all possible tours, then it has to examine about N! tours, which increases exponentially with N. Thus for a problem with fifty cities, the exhaustive search algorithm would need to evaluate 50! solutions. Let's say that evaluating one solution costs one nanosecond, then evaluating all possible solutions would cost about 10^{48} years, which is much longer than the age of the universe. Clearly, exhaustive search cannot be used for solving such combinatorial optimization problems and heuristic search algorithms have to be used that can find good solutions in a short time, although they do not always come up with the optimal solution. There is a large number of different meta-heuristic algorithms, such as tabu search, simulated annealing, multiple restart local hill climbing, ant colony algorithms, and genetic algorithms. Genetic algorithms differ from the others in the way that they keep a population of solutions and use recombination operators to form new solutions.

Evaluating one solution costs one nanosecond, then evaluating all possible solutions would cost about 10^{48} years, which is much longer than the age of the universe

Formal description of an optimization problem

Optimization problems consist of two components: the representation space and the evaluation (or fitness) function. The representation space denotes all possible solutions. For example, if we want to solve the TSP, the representation space consists of all possible tours which are encoded in some specific way. If we want to throw a spear at some target and can select the force and the angle to the ground, the representation space consists of two continuous dimensions that take on all possible values for the force and angle. Let's call the representation space S and a single solution s. The evaluation function (which in the context of evolutionary algorithms is usually called a fitness function) compares different solutions to each other. Although solutions could be compared on multiple criteria, let's assume for now that there is a single fitness function f(.) which maps a solution s to a specific fitness value f(s). The goal is to find the optimal solution s* that has the maximal fitness.

Finding a solution with local hill-climbing

Heuristic search algorithms usually start with one or more random solutions which are then evaluated. A simple, often used algorithm called local hill climbing starts with a random solution and then changes this solution slightly in some way. Then, this new solution is evaluated and if it is better than the previous one, it is kept and otherwise the previous one is kept. This simple process is repeated until the solution is good enough or a fixed amount of time has expired. The problem of this algorithm is that it often finds a local optimum. A local optimum is a solution which is not the global optimum (the best

solution in the representation space), but one which cannot be improved using the specific change operator. Thus, a local optimum is the best solution in a specific subspace (or attractor in the fitness landscape).

Genetic algorithms

In contrast to local hill climbing, genetic algorithms (GAs) use a population of individuals to search for solutions. The advantages of a population are that the search is done in a distributed way and that individuals are enabled to exchange genetic material. Searching with a population also allows for parallel computation, which is especially useful if computing the fitness of solutions takes a long time. For solving real world problems with genetic algorithms, such as a time-tabling problem which requires to schedule, for example, buses to drivers so that all buses have one driver and no driver has to drive when (s)he has indicated that (s)he does not want to drive, the question arises how to make a representation of the problem. This is sometimes more art than science and research has indicated that particular representations allow better solutions to be found much earlier. A genetic algorithm looks as follows in pseudo-code:

1. Initialize a population of N individuals
2.Repeat steps 2-9 until some termination condition holds:
3. Evaluate all individuals in the population using the fitness function
4.Repeat steps 5-8 N times:
5. Select two individuals for reproduction with a probability according to their fitness values
6. Recombine these two parent individuals to one new offspring
7. Mutate the offspring
8. Insert the offspring in a new population
9. Replace the population by the new population

Except for constructing a representation, we also need to find ways to initialize a population, to construct a mapping from genotype to phenotype (the genotype is the encoding in the chromosome on which the genetic operators work, whereas the phenotype is tested using the fitness function), and also to make a fitness function for evaluating an individual (different fitness functions would favor the same optimal solution,

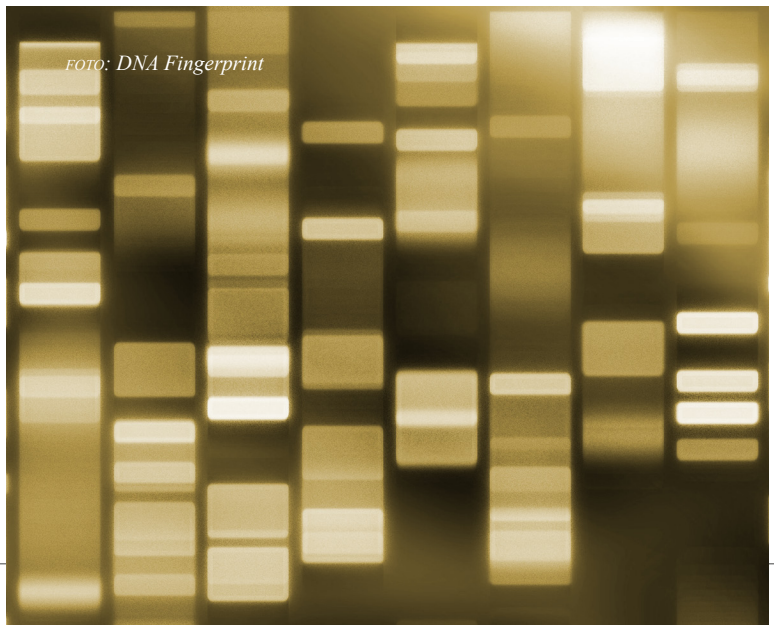


FOTO: DNA Fingerprint

but one of these can be more useful for the genetic algorithm to find it). There are also more specific steps: we need to design a mutation operator, a recombination operator, we have to determine how parents are selected for reproduction, we need to decide how individuals are used to construct a new population, and finally we have to decide when the algorithm has to stop.

Memetic Algorithms

There is an increasing amount of research that combines GAs with local hill-climbing techniques. Such algorithms are known as memetic algorithms. Memetic algorithms are inspired by memes (Dawkins,1976), pieces of mental ideas, like stories, ideas, goals, and gossip, which reproduce (propagate) themselves through a population of meme carriers. Similar to the selfish gene idea, in this mechanism each meme uses the host (the individual) to propagate itself further through the population. In this way it competes with different memes for the limited resources (there is always limited memory and time for knowing and telling everything). The difference between genes and memes is that the first are inspired by biological evolution and the second by cultural evolution. Cultural evolution is different because Lamarckian learning is possible in this model. That means that each transmitted meme can be changed according to receiving more information from the environment. This makes it possible to optimize each different meme before it is transmitted to other individuals. To construct a memetic algorithm, we can combine genetic algorithms with a simple local hill-climber. When a local hill-climber is used, each individual is not truly optimized, but only brought to its local maximum. If it would be possible to fully optimize the individual, we would not need a genetic algorithm at all. Memetic algorithms have been compared to GAs on a number of combinatorial optimization problems and experimental results indicated that the memetic algorithms found better solutions than standard genetic algorithms.

Discussion

There are now many conferences in the field of evolutionary computing and meta-heuristics. During the previous two decades many new algorithms have been developed for real-coded search spaces. These new methods include differential evolution, particle swarm optimization, and free search. There are also extensions to multi-objective problems where all solutions should be returned, which are not equal or worse in all objectives compared to some other solution. This set of incomparable solutions is called the Pareto optimal set and finding the whole Pareto set is much harder than finding a single optimal solution. To conclude, evolutionary computing is a very interesting field full of different algorithms. There are also many applications such as in logistics and transport, bio-informatics, game-playing, artificial art, and scheduling sport games.

Memetic algorithms are inspired by ideas, goals, and gossip, which reproduce themselves through a population of meme carriers.

PUZZEL

MAKER: Eric Jansen

Onderstaand diagram is een zogenaamde binaire puzzel, de naam zou de gemiddelde Informaticus of KI'er wel aan moeten spreken. De regels van de puzzel zijn als volgt:

1. Elke cel moet een 0 of een 1 bevatten.

2. Er mogen niet meer dan twee dezelfde cijfers direct naast elkaar of direct onder elkaar staan.

3. Elke rij en elke kolom moet evenveel nullen als enen bevatten.

4. Elke rij is uniek en elke kolom is uniek. Een willekeurige rij mag echter wel hetzelfde ingevuld worden als een willekeurige kolom.

5. Elke binaire puzzel heeft een unieke oplossing. Deze oplossing kan altijd gevonden worden zonder te gokken.

Stuur de oplossing van de puzzel vóór 1 november naar brainstorm@svcover.nl en maak kans op een leuke prijs!

1													
							0			0			
			0			0	0					0	
					0				0			1	
			1								0		
		1	1				0				0	0	
				0				1					
								1					1
1			0				0				0		
1			1		1			1	1		0		
													0
				0				1					0
	0		1				0					1	
									0	0		1	

HOE COVER

AUTEURS: *Anita Drenthen en Mart van de Sanden*

Cover heeft de afgelopen 10 jaar behoorlijk veel veranderingen meegemaakt. Soms zijn het veranderingen die meer komen van buitenaf, maar soms zijn het natuurlijk ook veranderingen uit het krijgen van een nieuwe stroom mensen die ieder hun eigen, nieuwe visie hebben en deze doorvoeren zodra ze aan de macht zijn (lees: bestuur doen). Om de mooie geschiedenis die Cover rijk is niet verloren te laten gaan bij de huidige leden en om oud-leden een kijkje te geven in de huidige situatie, doen wij een poging tot het beschrijven van een stukje geschiedenis. Hierbij beginnen we rond 2001, de tijd dat Mart begon met studeren en Cover nog CoVer heette, tot en met nu. Vanaf 2006 zal Anita het vooral verder vertellen, aangezien zij toen bij Cover kwam en een aantal jaren later actief het bestuur in hopte. Ook heeft Joël nog geholpen met de precieze invulling van de CoVer-Cover-transformatie, aangezien hij toen in het bestuur zat en dus nog het beste weet hoe dat verliep. We hopen samen in dit stukje wat leuke herinneringen op te halen en een beetje inzicht te geven in hoe Cover veranderd is in de afgelopen jaren.

Het tijdperk voor het tijdperk voor het tijdperk van het CoVer-huis

Toen Mart KI begon te studeren in 2001, zat CoVer ergens in een kamertje op de derde verdieping van het PPSW-gebouw In dit jaar was KI voor het eerst geen bovenbouwstudie meer en hadden we een jaar lang een bovenbouw- en een 5-jarige-opleiding tegelijkertijd. Dit betekende opeens veel meer leden; CoVer groeide uit van een extreem actieve doch kleine groep mensen, naar een extreem actieve en veel grotere groep mensen. Bijna iedereen kende bijna iedereen en commissies waren groot. Het was zeker een tijdperk van verandering, want CoVer is twee keer verhuisd in een korte tijd toen. Eerst naar een huisje in de hortustuin achter (toentertijd) PPSW, wat nu GMW heet, en direct daarna, in hetzelfde studiejaar nog, verhuisde CoVer weer met de opleiding naar de Grote Appelstraat.

Het tijdperk voor het tijdperk van het CoVer-huis

We gaan direct verder naar de jaren daarna; het tijdperk van de LiftCie, klaverjassen, de CultCie en nog veel meer. De excursie naar Noorwegen kan daarbij natuurlijk niet ontbreken! Vooral het vier dagen in een hutje in 'the middle of nowhere' zitten was erg bijzonder. Om een beeld te geven: er was geen douche of WC, maar er zat een gat in de grond en er was een HEEL koud beekje om je te wassen. We gingen verder met z'n allen in de sauna, vuurtje stoken, houthakken, jongleren en frisbeeën (ja, volgens ons is die traditie hier begonnen!). Ook het feit dat we hier een nationale Noorse feestdag konden vieren was natuurlijk erg gaaf. En als laatste, zoals kenmerkend bij elke reis

Cover WERD

die CoVer of Cover georganiseerd heeft, viel de helft bijna in slaap bij de praatjes van universiteiten. Erg gênant, maar wel kenmerkend voor hoe gezellig studiereisjes zijn!

Verder was de Introductiecommissie die het kamp van 2003 organiseerde een extreem actieve commissie. Zo actief, dat we (Mart zat uiteraard in deze commissie) besloten om zelfs meer te organiseren dan alleen een introductiekamp. Dit is waar de CultCie geboren is. Het thema was "paranoia", maar iedereen kent het als "het geheim van de lamb" met het logo van een paranoia schaap. We namen paranoia zo letterlijk dat we zelfs het thema geheim gehouden hebben! We vergaderden elke dinsdagavond in Cafe de Koster waar we een keer in één vergadering 106 euro bier hebben weggedronken. We maakten schaduwnotulen zodat het bestuur er niet achter kwam wat het echte thema was, we hebben zelfs een keer een nepvergadering gehouden toen de Commissaris Intern eens kwam meevergaderen en zijn toen zij wegging weer opnieuw begonnen met de echte vergadering.

Daarna was het opeens lustrum! CoVer bestond 10 jaar! Een hele week niets anders dan feest vieren, bijna niemand ging meer naar college. Er was een fietssloopwedstrijd, een stuk of drie feesten, waarvan één met een stripper (het thema was dan ook X(XX)) en ieder lid mocht ook een afdruk van z'n hand op de muur van de CoVer-kamer maken. Erg gaaf dus.

Als laatste van deze jaren was er ook nog de Amerika-reis, welke uiteraard ook briljant was! Konden we daar maar blijven, het was heel erg gezellig! Eerst Boston, toen naar Harvard, MIT en nog een paar universiteiten daar. MIT media lab was supercool: ze hadden Leonardo en nog veel meer robots, projecten met stereovisie en ze maakten menselijke huid na op een robotarm. Bij Harvard was vooral de meeting met de internationale studentenvereniging bijzonder. Het was in een dorm die er uitzag als de meest 'Harvard like dorm' die je ooit op TV gezien hebt, met eigen bedieningsstaf en een zwembad!

Het tijdperk van het CoVer-huis!

En dan komen we eindelijk bij het tijdperk waarbij CoVer een heel CoVer-huis had! Deze hadden we te danken aan de dieven die monitoren stalen op de Grote Appelstraat. De beveiliging werd opeens strenger en we konden minder goed na 21.00 uur film kijken in het KI-gebouw. Ter compensatie heeft Lambert voor ons een heel huis geregeld, vlakbij het Muntinggebouw, waar toen nog alle colleges waren! Aan het einde van het studiejaar 2003/2004 verhuisden we ernaartoe. Ondanks dat het zo ontzettend leuk klinkt, was het in dat jaar een moeilijke beslissing. De originele CoVer-kamer was altijd open en iedereen kwam daar ook de hele dag door gezellig zitten en kletsen.


Het bestuur hoefde er daarvoor niet te zijn, want de kamer had een apart kamertje waar de CoVer-computer, kas en andere belangrijke dingen achter slot en grendel konden. In het CoVer-huis kon je vervolgens alleen maar naar binnen als het bestuur er was en bijna niemand kwam meer even kort langs tussen colleges door. Ondanks deze moeilijke keuze is er toch voor het huis gekozen, waar we als CoVer zeker veel lol hebben gehad! Mart is er zelfs nu nog niet zo zeker van of het een goede keus is geweest, maar een gaaf verhaal voor de jongere Covergeneratie is het nog steeds: "Wist je dat Cover vroeger een HEEL HUIS had?!?!"

Ook in deze jaren waren er natuurlijk leuke excursies, zoals de excursie naar Schotland. Naar het mooie Edinburg en ook nog de hooglanden in! Ik (Mart) mis nog steeds Sullievans Wake, de geweldige lerse pub met live muziek in Edinburg, die helaas een paar jaar later volledig is afgebrand. Het was gezellig in Schotland, elke dag in groepjes samen koken in het hostel en veel spelletjes spelen en natuurlijk ook een hoop frisbeeën en zelfs volleyballen. De nachtelijke spooktoer door Edinburg was ook erg mooi. Het whiskymuseum was ook om nooit meer te vergeten!

Hierna werd het CoVer-huis helaas gesloopt en moesten we verhuizen naar een piepklein kamertje in het Idea-gebouw, achterop het Zerniketerrein. Een LACK tafel en twee bureau's en het was vol! Hierdoor was er even een dipje in activiteiten zoals filmavonden, waarbij er zelfs filmavonden zijn geweest waarbij maar twee mensen kwamen. De kamer stond simpelweg te ver weg van de studenten, die ook nog vaak college hadden in de binnenstad. Als ze al langskwamen in de kamer was er zo weinig ruimte dat je nooit lang bleef plakken. Ook geen ideale situatie dus.

CoVer werd Cover

Na de verhuizing naar het Idea begon het studiejaar 2006-2007, het jaar waarbij Mart met zijn generatie hun CoVer-piek al bereikt hadden en langzaam minder actief werden. Er was een nieuwe wind eerstejaars binnengekomen, waaronder Anita, die begonnen met het overnemen van het schip. Het enthousiasme van de oude groep en de liefde voor CoVer werd gelukkig nog wel doorgegeven aan de jongere generatie. Zij wisten dan ook dat ze deze vereniging zeker verder moesten laten bloeien! Het contact tussen deze generaties viel gelukkig niet helemaal weg, Mart was altijd (té?) gemakkelijk over te halen om commissies in te gaan en samen hebben we dan ook nog de FotoCie en de ExCie naar Ierland gedaan, waarbij de oude verhalen natuurlijk nog wel eens naar boven kwamen. Maar andersom hebben de meesten van de oude generatie de transformatie van CoVer naar Cover toch echt niet meegekregen. Voor hen zal Cover altijd CoVer blijven.

Over de transformatie valt veel te vertellen, maar we proberen het hier een beetje kort samen te vatten zodat er ook nog wat ruimte overblijft voor andere stukjes in deze Brainstorm. Het komt erop neer dat de indeling van de opleidingen veranderden: er kwamen drie poten in de Faculteit Wiskunde en Natuurwetenschappen waar alle opleidingen onder moesten vallen. Kunstmatige Intelligentie viel (uiteindelijk) samen met Informatica onder

#2, Jaargang 12, juni 2006

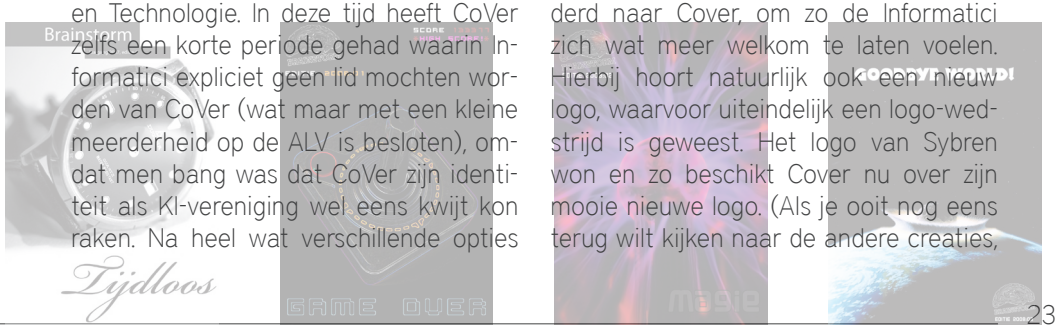
de poot 'Opleidingsinstituut Informatica en Cognitie'. De andere twee poten zijn Levenswetenschappen (voor Biologie, Life Science and Technology en Farmacie) en Natuurwetenschappen en Technologie (voor Wiskunde, Natuurkunde, Sterrenkunde, Scheikunde, etc.). In erse instantie was het idee om voor elke poot één vereniging te hebben. Bij Levenswetenschappen is dan ook een fusie ontstaan tussen de bestaande verenigingen, enkel Farmacie bleef zijn eigen vereniging houden. Bij Natuurwetenschappen en Technologie is er een nieuwe vereniging opgericht (ßasic), waarin uiteindelijk de oude verenigingen zouden moeten vallen (wat overigens uiteindelijk niet gebeurd is, de oude verenigingen bestaan nog steeds, maar dat ter zijde). Voor de FMF was deze opzet echter een groot probleem, aangezien zowel Informatici uit onze tak, als studenten van de asic-tak, lid waren van hun vereniging.

Wist je dat Cover vroeger een HEEL HUIS had?!?!

Op dit moment begon Cover dus met het overwegen van verschillende opties: onder andere fuseren met de FMF of enkel een mastervereniging worden voor KI. Of het hele OIC maar weer opheffen en KI bij Levenswetenschappen te stoppen en Informatica bij Natuurwetenschappen en Technologie. In deze tijd heeft CoVer zelfs een korte periode gehad waarin Informatici expliciet geen lid mochten worden van CoVer (wat maar met een kleine meerderheid op de ALV is besloten), omdat men bang was dat CoVer zijn identiteit als KI-vereniging wel eens kwijt kon raken. Na heel wat verschillende opties af te wegen (met behulp van een zeer goede werkgroep van CoVer die alles tot in de puntjes uitgezocht heeft), kwamen de Informatici uiteindelijk toch bij CoVer terecht en werd Cover dé vereniging voor de OIC-poot van de faculteit! Een keuze waar de huidige Coverleden nog steeds erg blij mee zijn. De sfeer is niet veel veranderd, Cover is alleen wat gegroeid en daarmee misschien nog wel gezelliger geworden!

Natuurlijk is er behoorlijk wat tijd overheen gegaan voordat we bij deze beslissing kwamen. Voordat we het vergeten, is er in de tussentijd nog de Eerstejaarscommissie in het leven geroepen! Super leuk idee natuurlijk, het is nog steeds een van de populairste commissies. En ook het NSVKI werd rond deze tijd opgericht: een landelijk overleg tussen de KI-verenigingen van Utrecht, Nijmegen en Groningen.

Dan kunnen we nu weer verder gaan met het jaar waarin de Informatici daadwerkelijk bij CoVer kwamen. Dingen moesten veranderen, we hadden een compleet nieuwe doelgroep erbij. Zo is CoVer in dit jaar ook toegetreden tot het SNiC (het landelijk overleg van Informaticaverenigingen waarbij jaarlijks een mooi symposium wordt neergezet) en is CoVer veranderd van naam! CoVer stond voor Cognitie Vereniging. De naam is veranderd naar Cover, om zo de Informatici zich wat meer welkom te laten voelen. Hierbij hoort natuurlijk ook een nieuw logo, waarvoor uiteindelijk een logo-wedstrijd is geweest. Het logo van Sybren won en zo beschikt Cover nu over zijn mooie nieuwe logo. (Als je ooit nog eens terug wilt kijken naar de andere creaties,



kijk dan eens bij de ALV-stukken van 19-02-2008)! Bij ons nieuwe logo hoorde ook een nieuwe kleur: rood! Hier werden ook direct stickers van gedrukt, om direct de nieuwe huisstijl overal in de stad maar achter te laten (en dan bedoelen we ook echt óveral!). Er is nu zelfs een Google-maps, waarin de leukste stickerplekken worden geprikt op de kaart. Moge Cover langzaam over de hele wereld verspreid worden! Een leuk verhaaltje hierbij is dat er vroeger ook enkele stickers waren met het oude logo en er nog enkele van over zijn. Een gedeelte van deze stickers is laatst bij de ledenbarbecue geveild en dat leverde enorm veel geld op!

Het contact met de FMF is hierbij ook bijzonder goed gebleven. Zij moesten natuurlijk wel hun positie van dé vereniging voor Informatici opgeven, wat natuurlijk zwaar kan komen te liggen. Nog steeds zitten er Informatici bij de FMF, dus werken we nog nauw samen. Onder andere hebben we samen vele LAN-party's en enkele Jongerejaars Colloquia georganiseerd en promoten we actief interessante activiteiten van elkaar. Vooral de gezamenlijke ritjes in de trein naar het SNiC en WISO zijn erg gezellig ook. We zijn dus blij dat het contact goed is gebleven.

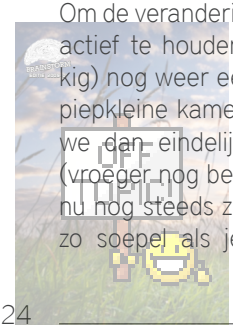
En daana?

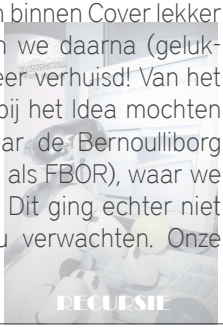
Om de veranderingen binnen Cover lekker actief te houden, zijn we daarna (gelukkig) nog weer een keer verhuisd! Van het piepkleine kamertje bij het Idea mochten we dan eindelijk naar de Bernoulliborg (vroeger nog bekend als FBOR), waar we nu nog steeds zitten. Dit ging echter niet zo soepel als je zou verwachten. Onze spullen hadden namelijk ongeveer twee maanden vertraging opgelopen in de verhuizing, wat uiteraard het nodige drama met zich meebracht. En toen de spullen uiteindelijk geleverd werden, waren onze banken en koelkast ook nog ineens verdwenen, aangezien ze niet voldeden aan de eisen van het gebouw! Na veel overleg zijn in ieder geval de banken weer teruggekomen, maar het was wel een rare periode, zo zonder je eigen spullen. Met de komst van onze nieuwe kamer is ook de SLACKcie opgericht, een commissie die het bestuur helpt met het openhouden van de kamer (onder andere voor tijdens bestuursvergaderingen). Een grandioos idee, nog steeds vragen we ons af waarom niet veel meer verenigingen dit doen!

Cover en CoVer staan dus, ondanks alle veranderingen, en dat zijn er nogal wat, nog steeds dicht bij elkaar.

En als laatste was in dit jaar natuurlijk ook de grote reis naar Ierland, georganiseerd door zowel Mart als Anita. Hier kunnen wij dus nog wel vijf pagina's over volschrijven, maar we zullen het maar op steekwoorden houden waar vele leuke herinneringen uit voort kunnen vloeien: Guinness, Groene kabouters met een pot vol goud aan het einde van de regenboog, het Maynooth-kasteel, de oude bibliotheek van Trinity College, het gigantische UCD campus, Johnny Bravo, ons promotiefilmpje (zoek op Youtube naar Cover excursie Ierland en promo!)... alles was AWESOME! Een mooie tijd om niet te vergeten.

In de jaren hierna vallen de veranderingen weer wat mee, er waren geen ver-



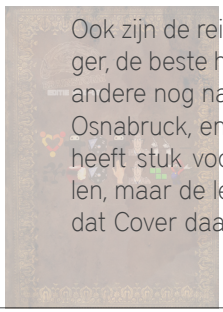
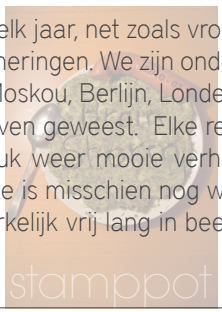


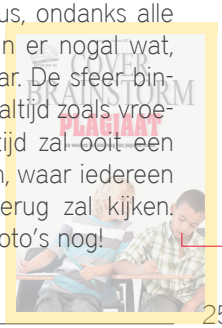
huizingen of naamsveranderingen of drastische veranderingen van buitenaf. Hierdoor hebben de besturen hierna veel meer aandacht kunnen besteden aan Cover nog mooier te maken dan Cover al was. Enkele van de leuke ideeën die zijn doorgevoerd zijn het starten van een eigen merchandiselijn, de oprichting van de Conditie (voor de nodige sportactiviteiten), de MeisCie (om ook de meiden van Cover zich meer thuis te laten voelen, met het organiseren van activiteiten die speciaal op die doelgroep gericht zijn) en de Promotie (voor het ontwerpen van mooie posters voor bijna elke activiteit). Al deze commissies blijven een succes en blijven ook gevuld. Dat is een mooi voordeel van iets meer leden: de actieve ledengroep is uitgegroeid naar ongeveer 55 mensen! Niet te veel om te vergeten wie wie is, maar ook niet te weinig om mooie dingen te organiseren. Ook is Cover actief geworden op social media om mensen een beetje meer inzicht te geven in het leven in een bestuursjaar, beschikken we nu over een Covervlag, vieren we nu jaarlijks onze verjaardag (Dies) en hebben we regelmatig 's middags een stafborrel om ook contact met de staf goed te onderhouden. Verder hebben we nu een nieuw boekhoudsysteem die de taken van de penningmeester een stuk fijner weten te maken en hebben we daadwerkelijk een nieuwe look op onze Cover-website!

Ook zijn de reisjes elk jaar, net zoals vroeger, de beste herinneringen. We zijn onder andere nog naar Moskou, Berlijn, Londen, Osnabruck, en Leuven geweest. Elke reis heeft stuk voor stuk weer mooie verhalen, maar de leukste is misschien nog wel dat Cover daadwerkelijk vrij lang in beeld kwam op het landelijk journaal van Moskou! Het was een nieuwsartikel over het alcoholprobleem in Moskou. Niet omdat wij nou zoveel problemen veroorzaakten natuurlijk! Maar aangezien ze nét in het wodkamuseum gingen filmen terwijl wij daar ook op bezoek waren!

De laatste verandering in deze jaren waar we nog veel mensen over horen praten, is het feit dat het bestuur van Cover in mijn (Anita's) bestuursjaar pakken is gaan dragen. De oude generatie is hier nog steeds niet aan gewend. Hun idee is dat Cover daardoor brallerig is geworden en de verkeerde kant op gaat. Deze mensen kan ik echter geruststellen: de pakken zijn slechts voor de formele afspraken met bedrijven etc. en bevallen daar erg goed. Daarnaast is het fijn niet meer het enige bestuur te zijn die met truien naar een consititieborrel gaat, we werden steeds vaker raar aangekeken in deze situaties. Ook al is het meegaan met wat de rest doet, je haalt er ook weer veel goede contacten uit met verenigingen die ons nu een stuk serieuzer nemen. Inmiddels is het alweer normaal en weten de eerstejaars niet eens dat het ooit misschien anders was. En oude generatie, wees gerust, op alle andere momenten loopt het bestuur nog steeds gewoon rond in een trui of een bestuurspolo en we zijn alles behalve brallerig geworden.

Cover en CoVer staan dus, ondanks alle veranderingen, en dat zijn er nogal wat, nog steeds dicht bij elkaar. De sfeer binnen de vereniging is nog altijd zoals vroeger en ook de huidige tijd zal ooit een 'gouwe ouwe' tijd worden, waar iedereen met veel plezier naar terug zal kijken. Gelukkig hebben we de foto's nog!



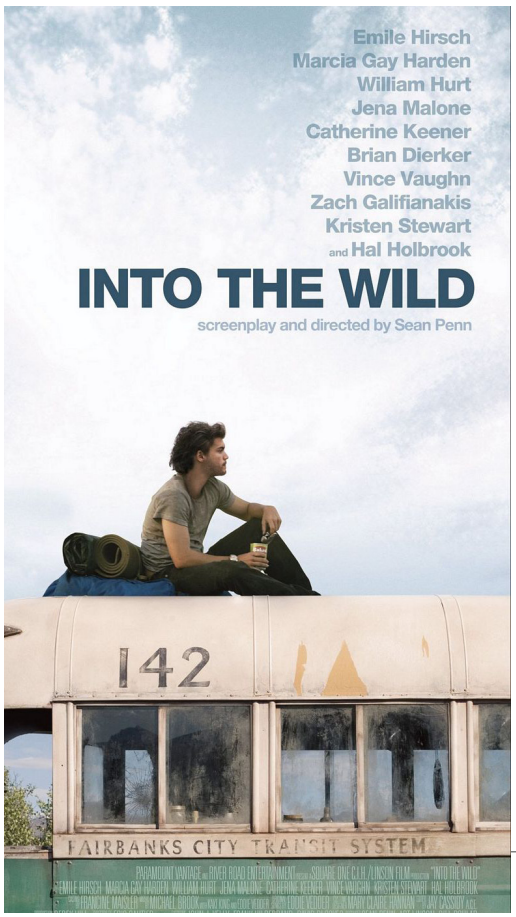
Switch

AUTEURS: *Sjors Lutjeboer / Sybren Römer*

In deze rubriek ruilen twee personen iets waar ze een bijzondere band mee hebben. Deze keer ruilen Sybren Römer (derdejaars Kl'er) en Sjors Lutjeboer (redactielid) een film. Zouden de films bevallen?

Into The Wild

Sjors - Voor Sybren heb ik de film Into the Wild uitgezocht. Into the Wild gaat over een jongen die zijn burgerlijke leventje achterlaat en op reis gaat met als bestemming de wildernis van Alaska. Onderweg lift hij, werkt hij op een boerderij, kayakt door de Grand Canyon en slaapt onder de sterren. Uiteindelijk belandt hij in Alaska. Zonder doel struint hij daar in het onbekende. Op een gegeven moment vindt hij een oude schoolbus die door jagers als hut wordt gebruikt. Maar aangezien het winter is, is hij alleen. Met wat rijst en een wapen moet hij een paar maanden zien te overleven. Verklappen hoe het afloopt is niet leuk dus dat zal ik niet doen.



Ik heb zelf gebackpackt door Nieuw Zeeland, dus het reizen spreekt me erg aan. Gewoon alleen in de wildernis zijn is fantastisch, daar heb ik zelf ook ervaring mee. Daarom spreekt deze film me erg aan en heb ik hem uitgekozen voor deze rubriek.

Ook spreekt de film me aan omdat het gebaseerd is op een waargebeurd verhaal. Op de één of andere manier hebben waargebeurde verhalen altijd iets bijzonders, iets extra's. Daardoor is het laatste shot van de film ook het mooiste. Ga hem zien!

Sybren - Van Sjors kreeg ik de film Into the Wild. Dit gaat over een jongen die alles wat hij heeft opgeeft om een tijd in Alaska te verblijven en rondreist met

bijna geen bezittingen. Ooit met Engels hebben we wel een beginnetje gemaakt met de film, maar nooit afgekeken. Ik vond de film echt leuk om te kijken; een stuk beter dan ik me van het begin herinner van vroeger.

Aangezien ik zelf ook vaak op vakantie ben geweest in de middle of nowhere, Noord-Finland bijvoorbeeld, en dat altijd erg vet vond, snapte ik wel waarom degene op wie de film gebaseerd is dit deed. Toen ik begon met de film wist ik nog niet dat het gebaseerd was op een waargebeurd verhaal. Dit maakte het allemaal net iets boeiender dan het al was. Al met al was het mijn tijd zeker waard en kan hem iedereen die hem nog niet heeft gezien aanbevelen. Bedankt Sjors!

Snatch

Sybren - Voor Sjors wilde ik graag een film verzinnen die hij nog niet had gezien. De wat bekendere films vielen daarom eigenlijk direct al af. Ik weet nog dat ik een aantal jaar geleden de film Snatch erg vet vond en ik er bijna nooit iemand over heb horen praten. Daarom heb ik Sjors deze film gegeven.

Sjors - Sybren had mij Snatch toegewezen om te kijken. Ik zal eerlijk zeggen dat ik deze film een jaar of vijf geleden al eens gezien had. Een hoop herkende ik weer, een hoop was ik ook weer vergeten. In Snatch lopen er eigenlijk twee verhalen in de film. Het ene gaat over illegale bokswedstrijden en alle illegale praktijken die daar bij komen kijken. Het andere gaat over een gestolen diamant. Dit alles loopt dwars door elkaar heen gedurende de film en komt geniaal samen aan het einde.

De film is van dezelfde maker als Lock Stock and Two Smoking Barrels. Als je deze hebt gezien en goed vond dan moet je zeker Snatch ook eens gaan kijken. De sfeer is behoorlijk hetzelfde. Bovendien speelt Jason Statham in beide films een hoofdrol.

De film is geweldig chaotisch en loopt van de hak op de tak, maar dat stoort helemaal niet. Het hoort bij de film. Goed opletten is wel een vereiste. En zet de ondertiteling aan als Brad Pitt begint te praten: daar versta je geen klap van.

De film is naar mijn idee erg vermakelijk en net als in Lock Stock zit ook deze film vol met geweldige conversaties en one-liners. Een must om gezien te hebben. Ik zal geen clous verklappen, maar ik wil toch nog even meegeven: let op de hond!


Ik zocht een mooie start
en kreeg 3 weken Ohio.
Carlo Klerk, Sogetist sinds 2007.
Ben jij net als Carlo op zoek naar de ideale start van je carrière? Welkom bij Sogeti. Wij bieden jou als Young Professional een introductieprogramma waar je U tegen zegt. Inclusief een boeiende Business Course van 3 weken aan de Ohio University in de Verenigde Staten. Samen met Amerikaanse topwetenschappers werk je daar aan jouw persoonlijke, technische en zakelijke vaardigheden. Zo leer je niet alleen Sogeti goed kennen, maar ook jezelf. En leg je meteen de basis voor een carrière met onbegrensde mogelijkheden. Ga je mee? Kijk op werkenbijsogeti.nl
SOGETI
gaat voor jou.