Hoofdstuk IV

Codetheorie voor CD-spelers

Iwan Duursma en Ludo Tolhuizen

Wellicht heeft de lezer wel eens gemerkt dat de muziek op een CD (Compact Disc) met kleine krasjes en stofjes nog steeds perfect wordt weergegeven. Hiertoe spelen, naast optica en electronica, fouten verbeterende codes een belangrijke rol. Zonder fouten verbeterende codes zouden de eisen aan de kwaliteit van een CD zó hoog zijn, dat (massa) productie zo niet onmogelijk, dan toch wel heel kostbaar zou zijn. Ook bij hard-disks in computers en bij modems voor de telefoonlijn worden fouten verbeterende codes toegepast.

In de praktijk worden ook fouten detecterende codes toegepast. Een voorbeeld hiervan is de ISBN-code voor de registratie van boeken. Als van een ISBN-nummer één cijfer verkeerd wordt opgeschreven, of twee naast elkaar gelegen cijfers worden verwisseld, dan is het resulterende nummer geen "geldig" ISBN-nummer. Met andere woorden, veel voorkomende schrijffouten kunnen worden opgespoord. Er moet dan echter, zodra de fout is gedetecteerd, worden nagevraagd wat het bedoelde ISBN-nummer was: fouten worden niet door de code verbeterd, zoals dat bij de CD speler of de computer (gelukkig!) wel het geval is.

Inleiding

De wiskunde is bij uitstek geschikt om regels aan te geven voor het opslaan van informatie en voor het ontwerpen van algoritmen die verstoringen in de opgeslagen informatie herkennen en corrigeren. Eerst illustreren we aan de hand van een voorbeeld de principes van foutendetectie en foutencorrectie. Daarna gaan we in op de belangrijke familie van Reed-Solomon codes en geven we een fouten corrigerende algoritme voor deze codes. Tenslotte beschrijven we hoe de codes worden toegepast in CD spelers.

Fouten herkennen. We beschouwen het eenvoudige geval dat een alfabet van drie letters wordt gebruikt om negen verschillende woorden te creëren:

AA, AB, AC,
BA, BB, BC,
CA, CB, CC.

We gaan ervan uit dat ieder woord een vaste betekenis heeft, maar welke dat is, is hier van minder belang. Het opslaan of verzenden van informatie als tweeletter-woord is zeer efficiënt maar weinig betrouwbaar. Zodra ergens een fout optreedt, zeg CA komt over als BA, dan ontstaat er een misverstand. Dergelijke fouten zijn vaak niet te vermijden, maar we kunnen ons beschermen tegen de misverstanden. Een manier is om woorden te herhalen. Bij ontvangst van CB AB concluderen we dan dat er een fout is opgetreden. Onder de aanname dat er niet meer dan een fout is opgetreden volgt dat ofwel CB tweemaal is verzonden danwel AB. We kunnen woorden met precies een fout dus eenvoudig herkennen. Maar we hebben geen zekerheid over welke fout is opgetreden.

Fouten corrigeren. Meer in het algemeen geldt als vuistregel voor betrouwbare informatie uitwisseling: vermijd woorden die veel op elkaar lijken. Dat is een eenvoudig principe dat ook in gesproken taal opgang doet. Een eenvoudig voorbeeld van een fouten verbeterende code is het telefoon-alfabet. Als over een slechte telefoonlijn de naam "Maurik" moet worden gespeld, dan gebruiken we bij voorkeur complete woorden (bijvoorbeeld "Marie-Anna-Utrecht-Rudolf-Izaak-Karel") in plaats van de afzonderlijke letters ("M-A-U-R-I-K"). Verschillende woorden uit het telefoon-alfabet verschillen veel van elkaar. Het is duidelijk dat het weinig zin heeft om "Atrecht" en "Utrecht" beide in het telefoon-alfabet te gebruiken.

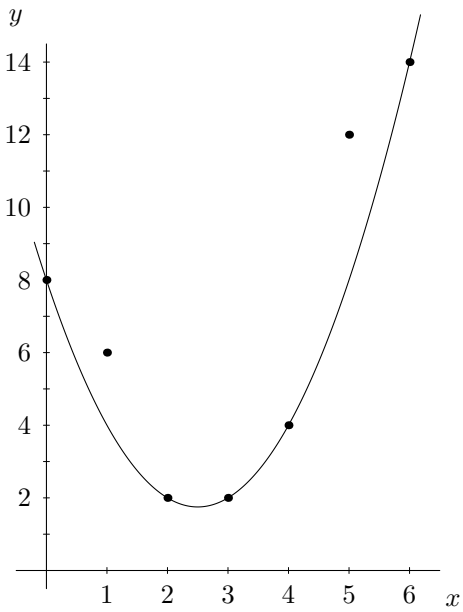
Het probleem wordt interessant als we korte woorden willen gebruiken (in verband met efficiency) die niet te veel op elkaar lijken (zodat kleine fouten niet tot misverstanden leiden). Dit wordt goed weergegeven door de volgende negen woorden

AA AA,	AB BB,	AC CC
BA BC,	BB CA,	BC AB
CA CB,	CB AC,	CC BA.

Net als bij herhaling van woorden gebruiken we nu vier letters, waar in principe de eerste twee letters volstaan. Maar in plaats van een simpele herhaling zijn de laatste twee letters nu zorgvuldig gekozen. Als klein wonder verschilt ieder tweetal woorden op precies drie plaatsen. Zo heeft ieder woord dat verschilt van AA AA bijvoorbeeld precies één letter A. Dit betekent dat we niet alleen fouten kunnen herkennen, zoals bij herhaling van woorden, maar ook kunnen corrigeren. Als we een woord met drie letters A ontvangen is duidelijk, onder de aanname dat er niet meer dan één fout is opgetreden, dat AA AA is verzonden. De kritische lezer verifiëert gemakkelijk dat bij gebruik van de negen woorden één willekeurige fout per woord is te herkennen en te corrigeren.

Reed-Solomon codes

Deze voorbeelden laten zien dat het voor foutencorrectie veel verschil maakt in welke vorm we informatie opslaan of verzenden. En de vraag rijst hoe we meer algemeen, voor gegeven woordlengte en alfabet, een verzameling kunnen construeren van woorden die onderling veel verschillen. Begin jaren zestig stelden I.S. Reed en G. Solomon [**1**] een constructie voor die in de praktijk zeer



FIGUUR 1. De unieke parabool door vijf punten

succesvol is gebleken. In de loop der jaren zijn er verschillende beschrijvingen gegeven van hun resultaat, vaak gemotiveerd door nieuwe toepassingen of nieuwe inzichten. Maar het achterliggende idee blijft onveranderd en we zullen het dadelijk formuleren als stelling.

Ter illustratie beschouwen we eerst Figuur 1. De geschetste kromme is de parabool met vergelijking $y = x^2 - 5x + 8$. Met de vergelijking vinden we enkele op de parabool gelegen punten:

$$(0, 8), (1, 4), (2, 2), (3, 2), (4, 4), (5, 8), (6, 14)$$

Omgekeerd bepalen de punten via interpolatie de vergelijking van de parabool. In feite hebben we aan drie punten genoeg.

LEMMA. *De vergelijking van een parabool is uniek bepaald door een willekeurig drietal erop gelegen punten.*

Het idee is nu als volgt. Als de zender genoeg op de parabool gelegen punten verstuurt naar de ontvanger, dan kan de laatste de vergelijking van de parabool terugvinden zelfs als niet alle punten correct worden ontvangen. Dit idee is bruikbaar bij verzending van een willekeurig drietal waarden a, b, c. De verzender gebruikt dan de vergelijking $y = ax^2 + bx + c$ om een aantal punten te berekenen en verstuurt vervolgens de punten. De ontvanger berekent als vergelijking door de punten $y = ax^2 + bx + c$ en vindt zo de relevante waarden a, b, c. De parabool en de vergelijking treden op als hulpmiddel en zijn op zich irrelevant voor de zender en de ontvanger. Die zijn alleen geïnteresseerd in

uitwisseling van de waarden a, b, c, en wellicht slechts in de betekenis die ze in een eerder stadium aan de waarden a, b, c hebben toegekend.

Hoe werkt dit voor het drietal waarden $1, -5, 8$? Laten we aannemen dat de verzender de zeven boven berekende punten verzendt en dat deze abusievelijk worden ontvangen als

$$(0, 8), (1, 6), (2, 2), (3, 2), (4, 4), (5, 12), (6, 14)$$

De ontvanger berekent dat de beste benadering wordt gegeven door de parabool met vergelijking $y = x^2 - 5x + 8$ en dat in dat geval op twee posities een fout moet zijn opgetreden (Figuur 1). We merken nog op dat de zender en ontvanger de keuze van de x-coördinaten kunnen afspreken zodat het volstaat om de y-coördinaten van de punten (in de afgesproken volgorde) te versturen.

Is het mogelijk dat er een tweede benadering bestaat die eveneens door vijf van de zeven punten gaat? In dat geval hebben de twee benaderingen minstens drie punten gemeen. Maar volgens het lemma gaat er door drie gegeven punten een unieke parabool en dat sluit een tweede benadering uit. We concluderen dat, als er op niet meer dan twee posities een fout is opgetreden, de oorspronkelijke vergelijking bepaald is door de unieke parabool die minstens vijf van de zeven punten bevat. We generalizeren het gevonden resultaat als volgt.

STELLING 1. *Stel dat het woord* $(f(0), f(1), \ldots, f(n-1))$ *wordt verzonden, waarbij* $f(x)$ *een veelterm is van graad ten hoogste* $n-1-2t$*. Indien niet meer dan* t *waarden van* f *verkeerd worden ontvangen, dan is* f *door de ontvanger te reconstrueren als de enige veelterm* g *van graad ten hoogste* $n-1-2t$ *zodat* $g(i) = f(i)$ *voor alle* $i \in \{1, \ldots, n\}$ *met hoogstens* t *uitzonderingen.*

BEWIJS. Er zijn minstens $n - 2t = k$ verschillende X-waarden waar $f(X)$ en $g(X)$ overeenkomen. Maar dan heeft de veelterm $f-g$ van graad $\deg(f-g) < n-1-2t$ minstens $n-2t$ nulpunten. Daaruit volgt dat $f = g$. □

Voor $n = 7$ en $t = 2$ vinden we het voorbeeld in Figuur 1 terug. We merken nog op dat, voor een gegeven f, het toevoegen van meer f-waarden aan een woord niet hetzelfde is als het simpel herhalen van eerder verzonden informatie. Ieder nieuw punt draagt op efficiënte wijze bij aan een grotere betrouwbaarheid van de te verzenden informatie. Hoeveel punten we toevoegen zal afhangen van de te verwachten foutenfrequentie.

Lineaire blokcodes

Heel algemeen is een code niets anders dan een verzameling woorden over een gegeven alfabet. Dat alfabet kan binair zijn (met als letters de twee symbolen 0,1), maar kan ook het gebruikelijke alfabet zijn (met letters A, B, ..., Z). Bij een blokcode hebben alle woorden dezelfde lengte n. Het aantal posities waarin twee willekeurige woorden verschillen ligt in het algemeen tussen 0 en n. Het minimum aantal posities waarin twee willekeurige maar verschillende woorden verschillen wordt de minimumafstand d genoemd. Een woord met t fouten kan op unieke wijze worden teruggevonden als $2t < d$. In dat geval is

er geen tweede woord dat op ten minste $n - t$ posities overeenkomt met het verstoorde woord. De code wordt dan een t fouten verbeterende code genoemd.

In plaats van het binaire alfabet gebruiken we ook wel het alfabet Z_p met letters $0, 1, \ldots, p-1$, waarbij p een priemgetal is. Net als bij het binaire alfabet kunnen we met de letters van dit alfabet optellen en vermenigvuldigen. Als de som of het product verschilt van de letters $0, 1, \ldots, p-1$ (dat wil zeggen, de uitkomst is groter dan $p-1$), dan vervangen we de uitkomst door de rest bij deling door p. We zeggen dan dat we rekenen modulo p. Een lineaire blokcode met alfabet Z_p is een blokcode met de eigenschap dat de som van twee codewoorden weer een codewoord is. De eerder beschreven 1-fout verbeterende code is lineair over het alfabet Z_3 met letters $0, 1, 2$.

00 00,	01 11,	02 22
10 12,	11 20,	12 01
20 21,	21 02,	22 10

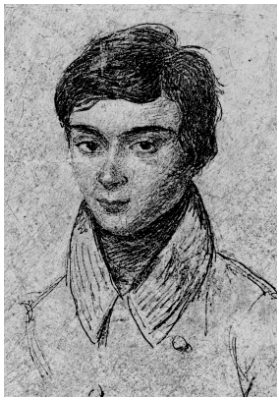
De codewoorden 2102 en 0222 hebben som 2324, wat na rest modulo drie het codewoord 2021 geeft. De hele code is volledig bepaald door de twee codewoorden 0111 en 1012. De overige codewoorden volgen door herhaald optellen. In het algemeen kunnen we dit formuleren door te zeggen dat een lineaire code een vectorruimte is die wordt opgespannen door geschikt gekozen basisvectoren. Het aantal benodigde basisvectoren wordt de dimensie k van de code genoemd. Een lineaire code van dimensie k over het alfabet Z_p heeft p^k codewoorden. De lengte n, dimensie k en minimumafstand d zijn de belangrijkste parameters van een lineaire code. We noemen de code van type $[n, k, d]$. De code in het voorbeeld is van type $[4, 2, 3]$ over het alfabet Z_3.

In Stelling 1 denken we bij de keuze van de X- en Y-coordinaten in eerste instantie aan de reële getallen. Maar de constructie blijkt ook te werken als we overschakelen op een ander alfabet.

In hun oorspronkelijke artikel definiëren Reed en Solomon codes over het alfabet $GF(2^m)$ waarin woorden van m nullen en enen als 'letters' optreden, voor een vastgekozen m. Om nu bijvoorbeeld de veelterm X^2+X uit te rekenen in de X-waarde 1001 is het nodig het alfabet van een vermenigvuldiging en een optelling te voorzien. Sinds de 19e eeuw en met name door het werk van de Franse wiskundige Evariste Galois (1811-1832) is bekend dat deze operaties op natuurlijke wijze zijn te definiëren voor het alfabet $GF(2^m)$, zodanig dat aan alle gebruikelijke axioma's wordt voldaan. Daarmee is het alfabet $GF(2^m)$ uitermate geschikt voor digitale opslag en manipulatie van gegevens.

Voor het formuleren van een procedure voor de correctie van fouten van een Reed-Solomon code vermijden we het alfabet $GF(2^m)$ met zijn minder intuïtieve optelling en vermenigvuldiging en wijken we uit naar het alfabet Z_p met de vertrouwde optelling en vermenigvuldiging. De lezer die vertrouwd is met de theorie van eindige lichamen ziet direct dat de procedure met minieme aanpassing geschikt is voor het alfabet $GF(2^m)$.

De Reed-Solomon constructie in Stelling 1 werkt zonder problemen over het alfabet Z_p voor een priemgetal p. Bij de keuze van de verzameling X-waarden zijn we daarbij beperkt tot $Z_p = \{0, 1, \ldots, p-1\}$. In het bewijs is het verder



Evariste Galois (1811-1832)

van belang op te merken dat ook over Z_p een veelterm niet meer nulpunten heeft dan zijn graad. De laatste eigenschap geldt omdat p een priemgetal is. Over Z_6, bijvoorbeeld, geldt dat $(X-1)(X-2)$ behalve $X=1$ en $X=2$ ook nulpunten heeft voor $X=4$ en $X=5$ en de graad is niet langer een bovengrens voor het aantal nulpunten. Voor foutencorrectie maken we gebruik van de volgende eigenschap.

Lemma. *Zij p een priemgetal en zij f een polynoom van graad ten hoogste $p-2$. Dan geldt dat de som $S(f)=f(0)+f(1)+\cdots+f(p-1)$ deelbaar is door p.*

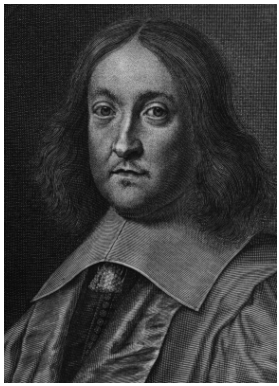
Bewijs. We schetsen het bewijs. Merk eerst op dat het voldoende is de speciale gevallen $f=1,X,X^2,\ldots,X^{p-2}$ te beschouwen. Voor $f=1$ neemt de som de waarde p aan. Voor het geval $f=X^b$ kiezen we een $a\neq 0$ zodanig dat a^b een rest modulo p ongelijk 1 heeft. Zo'n getal a bestaat omdat het aantal nulpunten van het polynoom X^b-1 in Z_p hoogstens $b<p-1$ is. We claimen dat de som met $f=X^b$ en de som met $f=(aX)^b$ dezelfde rest hebben modulo p. Er geldt namelijk dat de termen in de tweede som dezelfde rest hebben als de termen in de eerste som op de volgorde van de termen na. Er volgt dat p een deler is van het verschil

$$S((aX)^b)-S(X^b)=S(a^bX^b)-S(X^b)=(a^b-1)S(X^b).$$

En omdat p geen deler is van a^b-1 deelt p de som $S(X^b)$.

In het bewijs is het van belang dat we a kunnen kiezen zodanig dat a^b rest ongelijk 1 heeft modulo p. Voor $f=X^{p-1}$ is het lemma niet waar, precies omdat er dan niet zo'n a bestaat. Dit is een oud resultaat, opgemerkt door Fermat, dat bekend staat als de kleine stelling van Fermat. We vermelden het hier zonder bewijs.

Kleine stelling van Fermat. *Voor een priemgetal p en een geheel getal a, niet deelbaar door p, geldt dat a^{p-1} een rest modulo p heeft gelijk aan 1.*



PIERRE DE FERMAT (1601-1665)

We illustreren het gebruik van het lemma voor Reed-Solomon codes met $p = 7$. Neem $f(x) = x^2 - 5x + 8$ met codewoord

$$(f(0), f(1), \ldots, f(6)) = (8, 4, 2, 2, 4, 8, 14) = (1, 4, 2, 2, 4, 1, 0) \pmod 7.$$

De som van de letters is inderdaad deelbaar door 7. Voor het ontvangen woord

$$(8, 6, 2, 2, 4, 12, 14) = (1, 6, 2, 2, 4, 5, 0) \pmod 7$$

is de som van de letters niet langer deelbaar door zeven en dit is een eerste teken dat er een fout is opgetreden. In het bijzonder kan het lemma worden gebruikt om fouten te herkennen.

Een algoritme voor foutencorrectie

We keren terug naar de Reed-Solomon codes. De stelling voor het coderen geeft een efficiënte en betrouwbare manier om informatie op te slaan of te verzenden. Als er niet meer dan t fouten optreden in een woord dan zijn deze in principe te corrigeren. Het probleem van de ontvanger is om voor een ontvangen woord een interpolatie te verrichten door een gegeven aantal punten, waarbij het hem niet bekend is welke punten betrouwbaar zijn en welke niet. In Figuur 1 wordt de juiste parabool alleen gevonden als de twee foute punten niet bij de interpolatie worden betrokken. In het algemeen blijkt het mogelijk de punten waar een fout is opgetreden te berekenen (mits het totaal aantal fouten niet groter is dan t).

DEFINITIE (SYNDROMEN). *Voor een woord met p letters $Y_0, Y_1, \ldots, Y_{p-1}$ uit Z_p, definiëren we $S_0 = Y_0 + Y_1 + \cdots + Y_{p-1}$, en voor $b > 0$ zetten we*

$$S_b = Y_1 + 2^b Y_2 + \cdots + (p-1)^b Y_{p-1}.$$

Voor een codewoord $f(0), f(1), \ldots, f(p-1)$, met f een veelterm, is S_0 niets anders dan $S(f)$ en $S_b = S(X^b f)$.

LEMMA. *Voor een priemgetal p en voor een veelterm f van graad ten hoogste $p-2t-1$ geldt dat $S_0, S_1, \ldots, S_{2t-1}$ allen deelbaar zijn door p.*

BEWIJS. De veelterm $X^b f$ in de som S_b is van graad ten hoogste $b+(p-2t-1) \leq (2t-1)+(p-2t-1) = p-2$ en we mogen het eerdere lemma toepassen.

Voor $p=7$, passen we het lemma toe op het codewoord $f(0), f(1), \ldots, f(6)$, waarbij $f(x) = x^2 - 5x + 8$. Het woord bestaat uit de letters $1,4,2,2,4,1,0$ en we vinden $S_0 = 0, S_1 = 0, S_2 = 0, S_3 = 0$ (modulo 7). Voor het ontvangen woord $1,6,2,2,4,5,0$ vinden we $S_0 = 6, S_1 = 1, S_2 = 4, S_3 = 5$ (modulo 7).

Neem nu aan dat bij het opslaan van een codewoord onbekende fouten $E_1, E_2, \ldots, E_t$ zijn opgetreden in de onbekende posities $X_1, X_2, \ldots, X_t$, respectievelijk. Met andere woorden, in de positie X_i treffen we in plaats van $f(X_i)$ de letter $f(X_i) + E_i$ aan. Met het lemma zien we dat de termen $f(X_i)$ geen bijdrage leveren (modulo p) aan $S_0, S_1, \ldots, S_{2t-1}$. Alleen de fouten E_i dragen bij en de syndromen van het ontvangen woord zijn te schrijven als

$$\begin{array}{rcccccc} S_0 & = & E_1 & + & E_2 & + \cdots + & E_t \\ S_1 & = & E_1X_1 & + & E_2X_2 & + \cdots + & E_tX_t \\ \vdots & & \vdots & & \vdots & & \vdots \\ S_{2t-1} & = & E_1{X_1}^{2t-1} & + & E_2{X_2}^{2t-1} & + \cdots + & E_t{X_t}^{2t-1} \end{array}$$

De syndromen in het linkerlid worden gegeven door de definitie en kunnen we uitrekenen. De waarden van de posities X_i en de foutenbijdragen E_i zijn onbekend. Zij

$$\sigma(X) = (X - X_1)(X - X_2)\cdots(X - X_t) = X^t + \sigma_1 X^{t-1} + \ldots + \sigma_{t-1}X + \sigma_t$$

het polynoom met nulpunten in de onbekende foutenposities. Dit polynoom vemenigvuldigen we achtereenvolgens met $1, X, X^2, \ldots, X^{2t-1}$, en we sommeren voor elk de waarden aangenomen in de X_i. Met de bovenstaande relaties zien we dan dat

$$(*) \qquad \begin{array}{rcccccc} S_t & + & \sigma_1 S_{t-1} & + \cdots + & \sigma_{t-1}S_1 & + & \sigma_t S_0 = 0 \\ S_{t+1} & + & \sigma_1 S_t & + \cdots + & \sigma_{t-1}S_2 & + & \sigma_t S_1 = 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ S_{2t-1} & + & \sigma_1 S_{2t-2} & + \cdots + & \sigma_{t-1}S_t & + & \sigma_t S_{t-1} = 0 \end{array}$$

Dit vatten we op als een stelsel van t vergelijkingen waaruit de t onbekenden $\sigma_1, \sigma_2, \ldots, \sigma_t$ zijn op te lossen. Voor het voorbeeld met $S_0 = 6, S_1 = 1, S_2 = 4, S_3 = 5$ vinden we de twee vergelijkingen

$$4 + \sigma_1 + 6\sigma_2 = 0 \qquad 5 + 4\sigma_1 + \sigma_2 = 0$$

met als oplossing $\sigma_1 = 1, \sigma_2 = 5$ en

$$\sigma(X) = X^2 + X + 5 = (X-1)(X-5) \pmod 7.$$

De waarden $X_1 = 1$ en $X_2 = 5$ zijn inderdaad de waarden waar de fouten zijn opgetreden (Figuur 1).

STELLING. *Zij $Y_0, Y_1, \ldots Y_{p-1}$ een ontvangen woord met foutenbijdragen $E_1, E_2, \ldots, E_t$ in de onbekende posities $X_1, X_2, \ldots, X_t$, respectievelijk. De coefficiënten van het polynoom $\sigma(X) = (X - X_1)(X - X_2)\cdots(X - X_t)$ worden gegeven door de oplossing van het stelsel vergelijkingen $(*)$.*

Bovenstaande procedure werd voor het eerst aangegeven door D. Gorenstein, W.W. Peterson en N. Zierler. De belangrijkste bijdrage komt echter van E. Berlekamp [**2**] en J. Massey [**3**], die een efficiënt algoritme beschrijven om het stelsel vergelijkingen op te lossen. Het succes van de Reed-Solomon codes hangt sterk samen met de buitengewone eigenschappen van dit algoritme.

Erasure decoding

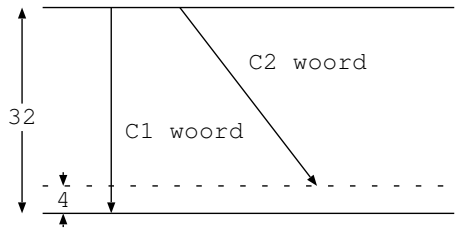
Stel dat we om een of andere reden denken dat sommige ontvangen symbolen waarschijnlijk fout zijn. We kunnen dan deze symbolen uitwissen en bij het decoderen niet gebruiken. Als we e symbolen uitwissen, dan gebruiken we de overblijvende $n - e$ symbolen om de veelterm f van graad ten hoogste $n - 1 - 2t = (n - e) - 1 - (2t - e)$ terug te vinden. Gebruiken we de stelling voor het coderen nu met $n - e$ in plaats van n en $2t - e$ in plaats van $2t$, dan volgt dat we in de $n-e$ posities $(2t-e)/2$ fouten kunnen verbeteren. Beschouw voor het voorbeeld met $n = 7$ en $t = 2$, het geval $e = 2$. Dat wil zeggen, er zijn twee verdachte symbolen uitgewist. Van de zeven ontvangen functie waarden gebruiken we alleen de resterende vijf waarden om de veelterm f van graad ten hoogste twee terug te vinden. Dit is mogelijk als er van de vijf waarden ten minste vier correct zijn. Als er drie fouten zijn opgetreden en twee daarvan kunnen we om een of andere reden aanwijzen en uitwissen, dan wordt f nog steeds goed teruggevonden. Zonder het uitwissen van symbolen zou dit niet mogelijk zijn en het heeft dus zin om verdachte symbolen uit te wissen. We zullen zien hoe dit wordt toegepast in de codes voor de CD speler.

CD spelers

CIRC codes. In CIRC[1], de fouten verbeterende code in het CD-systeem, werken twee eenvoudige codes samen. Beide codes zijn lineaire blokcodes. De ene code C_1 is een [32,28,5] code over GF(2^8). De andere code C_2 is een [28,24,5] code, ook over GF(2^8). In het alfabet GF(2^8) representeert iedere letter in een codewoord acht bits. De codes zijn van het type Reed-Solomon en kunnen beiden met een versie van het Berlekamp-Massey algoritme twee fouten per woord corrigeren. Deze twee codes worden gecombineerd tot een sterke code die kan worden gedecodeerd door de relatief eenvoudige decodeer-algoritmen voor C_1 en C_2 met elkaar te combineren. Het principe waarmee C_1 en C_2 in CIRC worden gecombineerd wordt geïllustreerd in Figuur 2. Codewoorden zijn de strips met alle kolommen in C_1 en alle (kortere) diagonalen in C_2. Bij het

[1] Cross Interleaved Reed-Solomon Codes

afspelen worden afwisselend kolommen en diagonalen gedecodeerd (met een decodeer algoritme voor C_1 respectievelijk C_2). Dit gebeurt zodanig dat alle symbolen die aan de C_2-decoder worden gevoed, al met behulp van C_1 zijn gedecodeerd. Door deze eerste decodeerslag is de foutenkans van een symbool al behoorlijk veel kleiner geworden, en die wordt dan door het C_2-decoderen nog veel kleiner.



FIGUUR 2. Het principe van de CIRC code

CIRC is niet alleen geschikt voor het decoderen van onafhankelijke fouten, maar ook voor het decoderen van gecorreleerde fouten, bijvoorbeeld krasjes op de CD. De bitvolgorde op de CD is zó dat een krasje fouten in één of meerdere opeenvolgende kolommen oplevert. Bij het C_2-decoderen staan de fouten uit een kolom verspreid over 28 C_2-woorden. Zijn er bijvoorbeeld twee kolommen vol fouten, en bevatten de andere kolommen hoogstens twee fouten, dan wordt alles nog goed gedecodeerd. Deze manier van verspreiden van gecorreleerde fouten over meerdere codewoorden staat in de literatuur bekend als "interleaving".

Foutenanalyse. Het is belangrijk om aan te geven hoe goed een fouten verbeterende code nu eigenlijk werkt. Eén seconde audio op CD bevat 1,41 miljoen bits. Een CD met een speelduur van een uur bevat dus $5{,}08\times10^9$ bits. Wil je, zeg, gemiddeld hoogstens een keer per uur niet goed decoderen, dan moet de bit-foutenkans hoogstens $1/(5{,}08\times10^9)$ zijn. Het is bijzonder tijdrovend om simulaties te doen die statistisch betrouwbare resultaten geven bij dergelijke zeldzame gebeurtenissen. Vaak werkt men met een foutenmodel en probeert dan met behulp van de kansrekening uitspraken te doen.

We geven een voorbeeld van een statistische analyse van CIRC. We nemen aan dat fouten in verschillende symbolen onafhankelijk van elkaar optreden en dat de foutenkans voor ieder symbool gelijk is aan p. Twee of minder fouten in een C_1-woord worden door C_1 goed gedecodeerd. De kans P_1 dat een symbool na C_1-decoderen fout is, voldoet aan

$$P_1 \approx \sum_{i=3}^{32} \binom{32}{i} \frac{i}{32} p^i (1-p)^{n-i}.$$

De formule is als volgt te begrijpen. Een foutpatroon met hoogstens twee foute letters wordt gecorrrigeerd. Een woord met minstens $i \geq 3$ fouten bevat na afloop meestal nog steeds i fouten, en dus heeft een symbool uit een foutpatroon

met i fouten (dat voorkomt met kans $\binom{32}{i}p^i(1-p)^{n-i}$) een kans van $i/32$ om fout te zijn.

Omdat met C_2 twee fouten kunnen worden gedecodeerd, is de kans dat een C_2-woord niet correct wordt gedecodeerd gelijk aan

$$P_2 = \sum_{i=3}^{28} \binom{28}{i} P_1^i (1-P_1)^{28-i}.$$

Bijvoorbeeld, als $p = 0{,}008$, dan $P_1 \approx 2{,}04 \times 10^{-4}$ en $P_2 \approx 2{,}77 \times 10^{-8}$. Per seconde staan er 7350 C_2-woorden op een CD. Een C_2-foutenkans van $2{,}77 \times 10^{-8}$ betekent dus dat er ruwweg om de $1/(7350 \cdot 2{,}77 \times 10^{-8}) \approx 4911$ seconden een fout optreedt, dus ruwweg eens in de 82 minuten.

De CIRC code is echter nog veel krachtiger dan hierboven aangegeven. We kunnen namelijk de resultaten van C_1 gebruiken om symbolen bij het C_2-decoderen uit te wissen. Eenvoudigheidshalve nemen we aan dat de C_1 decoder drie of meer fouten altijd detecteert. Symbolen van een C_1-woord dat niet kon worden gedecodeerd worden uitgewist bij het decoderen van C_2. De kans $\hat{P}_1$ dat een gegeven symbool in een C_2-woord uitgewist wordt, is dan

$$\hat{P}_1 = \sum_{i=3}^{32} \binom{32}{i} p^i (1-p)^{32-i}.$$

Het decodeer-algoritme voor C_2 vindt het goede antwoord bij vier of minder uitgewiste symbolen indien, zoals we hadden aangenomen, alle niet-uitgewiste symbolen correct zijn. Daarom is onder bovenstaande aanname de kans dat een C_2-woord niet goed wordt gedecodeerd gelijk aan

$$\hat{P}_2 = \sum_{i=5}^{28} \binom{28}{i} \hat{P}_1^i (1-\hat{P}_1)^{28-i}.$$

Als $p = 0{,}008$ vinden we $\hat{P}_1 \approx 2{,}13 \times 10^{-3}$ en $\hat{P}_2 \approx 4{,}13 \times 10^{-9}$. We zien dat $\hat{P}_2$ kleiner is dan P_2, wat het belang van het uitwissen van symbolen nog eens onderstreept.

De situatie is wat minder gunstig (en wat minder eenvoudig te analyseren) omdat een "gecorrigeerd" C_1-woord soms toch nog fouten bevat. Het uitwissen van symbolen uit woorden die niet door de C_1-decoder konden worden gecorrigeerd blijft echter gunstig om de foutenkans na decoderen te verkleinen.

Literatuur

1. E. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.
2. J.H. van Lint, *Coderingstheorie*, Colloquium discrete wiskunde **107** (1968), Amsterdam: Mathematisch Centrum,, 51–75.
3. J. Massey, *Shift register synthesis and BCH decoding*, IEEE Trans. Inform. Theory **15** (1969), 122-127.
4. I.S. Reed en G. Solomon, *Polynomial codes over certain finite fields*, J. Soc. Indust. Appl. Math. **8** (1960), 300-304.
5. J.H. Weber, *Foutenverbeterende codes*, collegedictaat, Delft: Technische universiteit Delft.