Les 7 Betrouwbaarheid en levensduur

7.1 Betrouwbaarheid van systemen

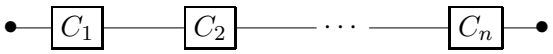
Als een systeem of netwerk uit verschillende componenten bestaat, kan men zich de vraag stellen hoe groot de kans is dat het systeem faalt. Dit hangt zeker van de kwaliteit van de componenten af, maar ook hoe deze componenten van elkaar afhangen en samengeschakeld zijn.

Voor een enkele component C noemen we de kans dat C (normaal) werkt de *betrouwbaarheid* van C. Als A de gebeurtenis is dat C werkt, dan is de betrouwbaarheid R (voor *reliability*) van C dus gegeven door de kans op A, dus door

$$R = P(A).$$

Rijschakeling

De eenvoudigste manier om verschillende componenten te combineren, is een *rijschakeling*. Hierbij wordt een opdracht achter elkaar van de componenten C_1, C_2, ..., C_n verwerkt, waarbij C_{i+1} pas begint als C_i klaar is. Het hele systeem werkt alleen maar als alle componenten werken.



Als we met A_i het normaal werken van de component C_i noteren en met A het goed functioneren van het hele systeem, geldt $P(A) = P(A_1, A_2, \ldots, A_n)$. Als de component C_i de betrouwbaarheid $R_i = P(A_i)$ heeft, dan is de betrouwbaarheid $R = P(A)$ van het hele systeem dus gegeven door

$$R = R_1 \cdot R_2 \cdot \ldots \cdot R_n = \prod_{i=1}^{n} R_i.$$

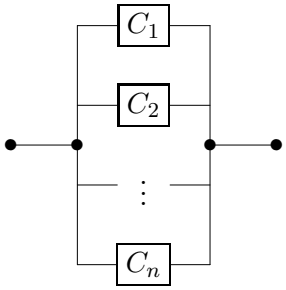
Merk op: We veronderstellen hierbij dat het falen van de enkele componenten onafhankelijke gebeurtenissen zijn.

Het is duidelijk dat zelfs bij zeer betrouwbare componenten de betrouwbaarheid van de rijschakeling met een groeiend aantal van componenten snel afneemt. Bijvoorbeeld hebben we voor $n = 10$ en $R_1 = \ldots = R_{10} = 0.99$ voor de rij van 10 componenten slechts nog een betrouwbaarheid van $R = 0.99^{10} \approx 0.904$.

Omdat systemen vaak uit heel veel componenten opgebouwd zijn en deze geen willekeurig hoge betrouwbaarheid kunnen hebben, wordt er vaak *redundantie* in een systeem ingebouwd. Dit betekent dat een systeem componenten bevat die overbodig zijn als alles goed werkt, maar die ervoor zorgen dat het hele systeem ook nog blijft werken als er een of meer componenten falen.

Parallelschakeling

De eenvoudigste vorm van redundantie is een *parallelschakeling*. Hierbij wordt voor het correcte functioneren van het systeem slechts een enkele van een aantal componenten benodigd.



Het hele systeem werkt als één van de componenten $C_1, \ldots, C_n$ werkt, dus is de kans $P(A^c)$ dat het hele systeem niet werkt gelijk aan de kans dat *geen* van de componenten werkt en dus gelijk aan het product van de kansen $P(A_i^c)$ dat de enkele componenten niet werken, dus $P(A^c) = (1-R_1)\cdot(1-R_2)\cdot\ldots\cdot(1-R_n)$. Voor de betrouwbaarheid $R = 1 - P(A^c)$ van het hele systeem geldt dus

$$R = 1 - (1 - R_1) \cdot (1 - R_2) \cdot \ldots \cdot (1 - R_n) = 1 - \prod_{i=1}^{n} (1 - R_i).$$

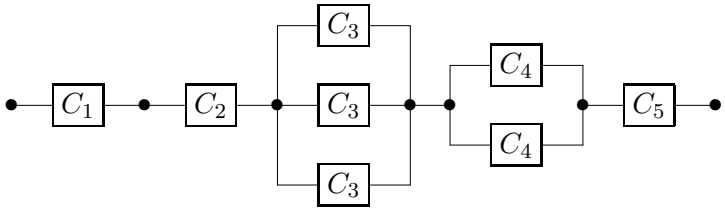
Ook hier gaan we weer ervan uit dat de componenten onafhankelijk van elkaar uitvallen.

Door een parallelschakeling kunnen we de betrouwbaarheid van een systeem snel verhogen, voor $n = 2$ en $R_1 = R_2 = 0.9$ hebben we bijvoorbeeld $R = 1 - (1 - 0.1)^2 = 0.99$. Echter hebben we hierbij de kosten verdubbeld om de betrouwbaarheid van 90% op 99% te verhogen.

Rij-parallel-schakeling

Als rij- en parallelschakelingen in een systeem gecombineerd worden, spreekt men van een *rij-parallel-schakeling*. Zo'n systeem krijgt men als men een rij- of parallelschakeling van een aantal componenten als een enkele component beschouwd en met dit soort componenten weer rij- en parallelschakelingen construeert. Door dit proces te herhalen, kan men redelijk ingewikkelde systemen realiseren. Tegelijkertijd levert dit proces ook het pad van de analyse van zo'n systeem, want door de stappen na te gaan kan ook de betrouwbaarheid van het systeem bepaald worden. We kijken hiervoor naar twee voorbeelden.

Voorbeeld 1: Een eenvoudig geval van een rij-parallel-schakeling is een rij van parallelschakelingen. In het volgende voorbeeld gaan we ervan uit dat de componenten in een parallelschakeling alle hetzelfde zijn.



De betrouwbaarheid R van het volledige systeem krijgen we als

$$R = R_1 \cdot R_2 \cdot (1 - (1 - R_3)^3) \cdot (1 - (1 - R_4)^2) \cdot R_5.$$

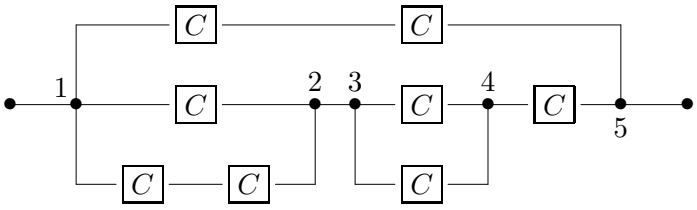
Als de componenten C_i bijvoorbeeld de betrouwbaarheden $R_1 = 0.95$, $R_2 = 0.99$, $R_3 = 0.7$, $R_4 = 0.75$ en $R_5 = 0.9$ hebben, geeft dit

$$R = 0.95 \cdot 0.99 \cdot (1 - 0.3^3) \cdot (1 - 0.25^2) \cdot 0.9 \approx 0.772.$$

Algemeen heeft een rij van s parallelschakelingen, waarbij de i-de parallelschakeling n_i componenten met betrouwbaarheid R_i bevat de betrouwbaarheid

$$R = (1 - (1 - R_1)^{n_1}) \cdot \ldots \cdot (1 - (1 - R_s)^{n_s}) = \prod_{i=1}^{s} (1 - (1 - R_i)^{n_i}).$$

Voorbeeld 2: We kijken naar een communicatienetwerk waarbij verschillende mogelijke paden tussen twee punten bestaan. De verbindingen op de paden bevatten zekere componenten waarvan het functioneren van de verbindingen afhangt, bijvoorbeeld versterkers voor GSM-signalen. Stel we hebben het volgende netwerk, waarbij de componenten C alle dezelfde betrouwbaarheid R hebben.



We zien dat het hele netwerk een parallelschakeling van het rechtstreekse pad $1-5$ en de verbinding $1-2-3-4-5$ is, waarbij $1-2-3-4-5$ een rij van eenvoudige parallelschakelingen is. De enkele betrouwbaarheden voor de deelverbindingen vinden we nu als volgt:

$$\begin{aligned}
R_{1-5} &= R^2 \\
R_{1-2} &= 1 - (1 - R)(1 - R^2) = 1 - (1 - R - R^2 + R^3) = R + R^2 - R^3 \\
R_{3-4} &= 1 - (1 - R)^2 = 1 - (1 - 2R + R^2) = 2R - R^2 \\
R_{1-2-3-4-5} &= R_{1-2} \cdot R_{3-4} \cdot R = (R + R^2 - R^3)(2R - R^2) \cdot R \\
&= 2R^3 - R^4 + 2R^4 - R^5 - 2R^5 + R^6 = 2R^3 + R^4 - 3R^5 + R^6.
\end{aligned}$$

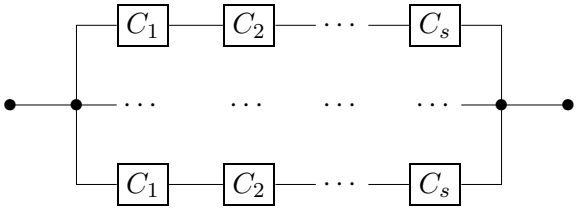
Als betrouwbaarheid R_{net} van het hele netwerk vinden we hieruit

$$\begin{aligned}
R_{net} &= 1 - (1 - R_{1-5})(1 - R_{1-2-3-4-5}) \\
&= 1 - (1 - R^2)(1 - 2R^3 - R^4 + 3R^5 - R^6) \\
&= 1 - 1 + 2R^3 + R^4 - 3R^5 + R^6 + R^2 - 2R^5 - R^6 + 3R^7 - R^8 \\
&= R^2 + 2R^3 + R^4 - 5R^5 + 3R^7 - R^8.
\end{aligned}$$

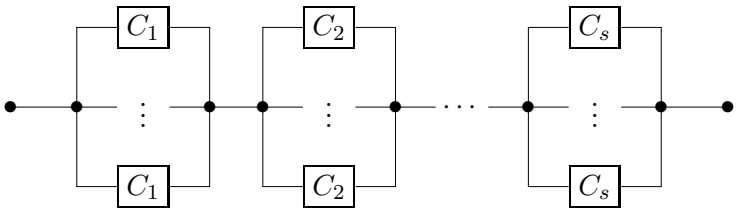
Als we $R = 0.9$ aannemen, krijgen we $R_{1-5} = 0.81$, $R_{1-2-3-4-5} \approx 0.874$ en $R_{net} \approx 0.976$. Voor $R = 0.99$ hebben we $R_{1-5} = 0.9801$, $R_{1-2-3-4-5} \approx 0.9897$ en $R_{net} \approx 0.9998$.

Als een systeem uit s stappen bestaat die achter elkaar uitgevoerd moeten worden, zijn er twee voor de hand liggende manieren om zo'n systeem met een *redundantiefactor* van m, d.w.z. met m componenten voor iedere stap, op te bouwen:

(1) Als parallelschakeling van m rijen die de s verschillende componenten bevatten.



(2) Als rijschakeling van s parallelschakelingen van telkens m componenten.



Als we de betrouwbaarheid van de component C_i met R_i noteren, vinden we voor de betrouwbaarheden van de versies (1) en (2):

$$R_{(1)} = 1 - (1 - R_1 \cdot R_2 \cdot \ldots \cdot R_s)^m = 1 - (1 - \prod_{i=1}^{s} R_i)^m,$$
$$R_{(2)} = (1 - (1 - R_1)^m) \cdot (1 - (1 - R_2)^m) \cdot \ldots \cdot (1 - (1 - R_s)^m)$$
$$= \prod_{i=1}^{s} (1 - (1 - R_i)^m).$$

In het speciaal geval dat alle componenten C_i dezelfde betrouwbaarheid R hebben, krijgen we voor de betrouwbaarheden van de twee versies eenvoudigere uitdrukkingen, namelijk:

$$R_{(1)} = 1 - (1 - R^s)^m \quad \text{en} \quad R_{(2)} = (1 - (1 - R)^m)^s.$$

Er laat zich aantonen dat altijd geldt dat

$$R_{(2)} = \prod_{i=1}^{s} (1 - (1 - R_i)^m) > 1 - (1 - \prod_{i=1}^{s} R_i)^m = R_{(1)},$$

d.w.z. de opzet (2) met de rijschakeling van parallelschakelingen heeft steeds een hogere betrouwbaarheid dan de opzet (1) met de parallelschakeling van rijschakelingen.

Deze belangrijke inzicht gaan we hier niet algemeen bewijzen, maar we gaan wel voor drie speciale gevallen na, dat het inderdaad zo is.

(A) $s = 2$, $m = 2$: Er geldt

$$R_{(1)} = 1 - (1 - R_1 R_2)^2 = 1 - (1 - 2R_1 R_2 + R_1^2 R_2^2) = 2R_1 R_2 - R_1^2 R_2^2.$$

Aan de andere kant is

$$\begin{aligned} R_{(2)} &= (1-(1-R_1)^2)(1-(1-R_2)^2) = (2R_1 - R_1^2)(2R_2 - R_2^2) \\ &= 4R_1R_2 - 2R_1^2R_2 - 2R_1R_2^2 + R_1^2R_2^2 \\ &= 2R_1R_2 - R_1^2R_2^2 + 2R_1R_2(1 - R_1 - R_2 + R_1R_2) \\ &= R_{(1)} + 2R_1R_2(1-R_1)(1-R_2). \end{aligned}$$

Omdat $R_i > 0$ en $1 - R_i > 0$ is dus inderdaad $R_{(2)} > R_{(1)}$.

(B) $s = 3$, $m = 2$, $R_1 = R_2 = R_3 = R$: Er geldt

$$R_{(1)} = 1 - (1-R^3)^2 = 1 - (1 - 2R^3 + R^6) = 2R^3 - R^6.$$

Aan de andere kant is

$$\begin{aligned} R_{(2)} &= (1-(1-R)^2)^3 = (1-(1-2R+R^2))^3 = (2R-R^2)^3 \\ &= 8R^3 - 12R^4 + 6R^5 - R^6 = 2R^3 - R^6 + 6R^3(1 - 2R + R^2) \\ &= R_{(1)} + 6R^3(1-R)^2. \end{aligned}$$

Uit $R > 0$ volgt dat $R_{(2)} > R_{(1)}$, want $(1-R)^2$ is sowieso positief.

(C) $s = 2$, $m = 3$, $R_1 = R_2 = R$: Er geldt

$$R_{(1)} = 1 - (1-R^2)^3 = 1 - (1 - 3R^2 + 3R^4 - R^6) = 3R^2 - 3R^4 + R^6.$$

Aan de andere kant is

$$\begin{aligned} R_{(2)} &= (1-(1-R)^3)^2 = (1-(1-3R+3R^2-R^3))^2 \\ &= (3R - 3R^2 + R^3)^2 = 9R^2 + 9R^4 + R^6 - 18R^3 + 6R^4 - 6R^5 \\ &= 3R^2 - 3R^4 + R^6 + 6R^2(1 - 3R + 3R^2 - R^3) \\ &= R_{(1)} + 6R^2(1-R)^3. \end{aligned}$$

In dit geval volgt uit $1 - R > 0$ dat $R_{(2)} > R_{(1)}$.

Andere typen van redundantie

Naast rij-parallel-schakelingen zijn er natuurlijk nog andere manieren om redundantie in een netwerk in te bouwen. Een mogelijkheid zijn zogeheten *m-uit-n* blokken, die uit n componenten bestaan en goed werken als minstens m van de n componenten normaal functioneren.

Als in een m-uit-n blok alle componenten betrouwbaarheid R hebben, heeft de hele blok de betrouwbaarheid

$$R_{m|n} = \sum_{k=m}^{n} \binom{n}{k} R^k (1-R)^{n-k}.$$

Het zou geen verrassing zijn dat hier de binomiale verdeling een rol speelt, want we moeten gewoon $k \geq m$ van de n componenten kiezen die goed werken.

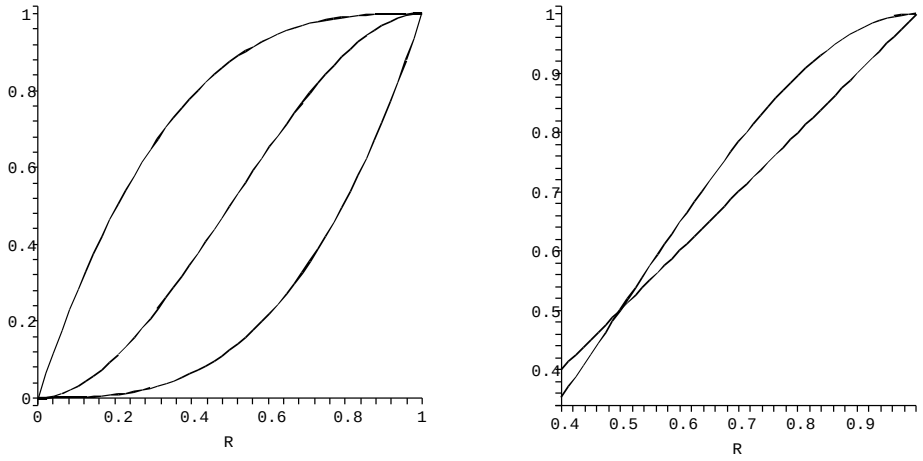
De kans voor het goed werken van k componenten is R^k terwijl de kans dat de resterende $n-k$ componenten niet werken $(1-R)^{n-k}$ is.

Als speciale gevallen van een m-uit-n blok vinden we voor $m=1$ de parallelschakeling en voor $m=n$ de rijschakeling terug.

Een belangrijk voorbeeld van een m-uit-n blok is een 2-uit-3 blok, die de naam *triple modular redundancy system*, afgekort TMR, heeft. Zo'n systeem heeft de betrouwbaarheid

$$R_{TMR} = R_{2|3} = R^3 + 3R^2(1-R) = R^3 + 3R^2 - 3R^3 = 3R^2 - 2R^3.$$

In het linkerplaatje van Figuur 11 zijn de betrouwbaarheden van een parallelschakeling, een TMR (2-uit-3 blok) en een rijschakeling met telkens 3 componenten als functies van de betrouwbaarheid R van een van de componenten te zien.



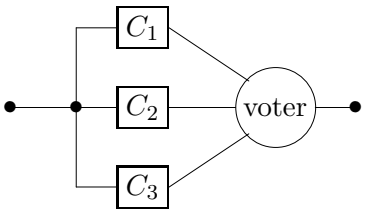
Figuur 11: Links: Betrouwbaarheden van m-uit-3 blokken voor $m = 1, 2, 3$. Rechts: Vergelijk van TMR met een van zijn componenten.

Voor een gegeven waarde van R heeft de parallelschakeling natuurlijk de hoogste betrouwbaarheid en de rijschakeling de laagste, de kromme voor de betrouwbaarheid van de TMR ligt tussen deze twee krommen.

Om de betrouwbaarheid van een TMR met een van zijn componenten te vergelijken moeten we kijken wanneer $3R^2 - 2R^3 > R$ is. Voor $R > 0$ is dit het geval als $3R - 2R^2 > 1$, dus als $2R^2 - 3R + 1 < 0$. De nulpunten van $2R^2 - 3R + 1$ kunnen we (bijna) gokken, er geldt $2R^2 - 3R + 1 = (2R-1)(R-1)$, dus zijn de punten waar $R_{TMR} = R$ geldt de waarden $R = 0$, $R = \frac{1}{2}$ en $R = 1$. Voor $\frac{1}{2} < R < 1$ geldt dat $R_{TMR} > R$ en voor $0 < R < \frac{1}{2}$ dat $R_{TMR} < R$.

Het rechterplaatje van Figuur 11 laat het verschil tussen de betrouwbaarheid van een TMR met drie componenten met betrouwbaarheid R en een enkele componenten met betrouwbaarheid R in afhankelijkheid van de waarde van R zien. Vooral voor hoge waarden van R geeft de TMR een duidelijk voordeel, bij $R = 0.9$ is zijn betrouwbaarheid bijvoorbeeld $R_{TMR} = 0.972$.

In de praktijk wordt een m-uit-n blok vaak door een *voter* gerealiseerd. De voter neemt gewoon een meerderheidsbeslissing over de resultaten van de enkele componenten.



Een probleem hierbij is dat bij componenten die alleen maar een keuze uit twee mogelijkheden hebben (ja-nee beslissing), een meerderheid van foutieve componenten tot een foutieve beslissing van de voter leidt. Daarom wordt meestal nog een detectie component achter de voter gezet, die deze gevallen moet herkennen. Maar omdat deze component ook niet perfect werkt, leidt dit tot een lagere betrouwbaarheid van de TMR.

Toepassing: Codering van bits

Een belangrijke toepassing van het verhogen van de betrouwbaarheid met behulp van redundantie is het versturen van digitale informatie. Als we een bit versturen, wordt slechts met een zekere kans p ook hetzelfde bit ontvangen, met kans $1-p$ wordt het bit door een storing omgeschakeld. Het versturen van een bit heeft dus de betrouwbaarheid p.

Om een hoge betrouwbaarheid te bereiken, kan men een bit n keer herhalen en dan een voter gebruiken. Het is slim om n oneven te kiezen, dus $n = 2m-1$, dan is bij minstens m correct ontvangen bits de meerderheid correct en we hebben dus een m-uit-$(2m-1)$ blok.

Het versturen van een boodschap door middel van herhaling noemt men een *repetition code*. De betrouwbaarheid van een repetition code met $2m-1$ herhalingen is

$$R = \sum_{k=m}^{2m-1} \binom{2m-1}{k} p^k (1-p)^{2m-1-k}.$$

Het is duidelijk dat een repetition code geen efficiënte manier is, om de betrouwbaarheid te verhogen, want zelfs voor een 2-uit-3 blok die niet zo'n grote verbetering geeft, moeten we al 3 keer zo veel bits versturen dan de boodschap eigenlijk bevat.

Een efficiëntere manier om digitale boodschappen met zekere redundantie te versturen zijn *foutherkennende codes* en *foutverbeterende codes*. Het idee hierbij is een boodschap op een slimme manier door verdere bits aan te vullen, bijvoorbeeld door *parity-check bits*.

Een parity-check bit werkt als volgt:

Als een boodschap van 7 bits verstuurd wordt, wordt de achtste bit op 0 gezet als de eerste 7 bits een even aantal 1en bevatten, en op 1 als de eerste 7 bits een oneven aantal 1en bevatten. De achtste bit is dus de (binaire) som van de eerste 7 bits. Als nu een boodschap ontvangen wordt, kan men weer

checken, of het laatste bit de som van de andere bits is. Als dit niet het geval is, is er minstens een bit veranderd, maar het is niet mogelijk te zeggen welke bit veranderd is. Als de som wel klopt, is de boodschap of correct ontvangen of er zijn minstens twee bits veranderd. Maar de kans hierop is al behoorlijk kleiner dan die op een enkel veranderde bit.

Omdat we niet kunnen zeggen welke bit veranderd is als de som niet klopt, heet een code met alleen maar een parity-check bit een *foutherkennende code*.

Maar hoe kunnen we met behulp van aanvullende bits fouten verbeteren? We zullen dit aan een voorbeeld bekijken, namelijk de $(7,4)$-Hamming code. Bij deze code wordt een boodschap van 4 bits aangevuld met 3 verdere bits die van de eerste 4 bits afhangen. Op deze manier zijn maar $2^4 = 16$ van de mogelijke $2^7 = 128$ woorden van 7 bits geldige codewoorden. De Hamming code wordt aangegeven door een matrix die als rijen de codewoorden voor de boodschappen 1000, 0100, 0010 en 0001 bevat. Deze matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We kunnen elke boodschap van 4 bits als som van de vier aangegeven boodschappen 1000, 0100, 0010 en 0001 krijgen, het bijhorende codewoord met 7 bits is dan de som van de overeenkomstige rijen van de matrix. Bijvoorbeeld hoort bij de boodschap 1010 het codewoord 1010101, de som van de eerste en derde rij van de matrix.

De cruciale eigenschap van de $(7,4)$-Hamming code voor het verbeteren van fouten is dat verschillende codewoorden van deze code steeds op minstens drie plekken verschillen. Het aantal verschillende bits van twee woorden (rijen van bits) noemt men ook de *Hamming-afstand* of kort *afstand* van de woorden. Als we nu een rij van bits ontvangen waarbij tegenover het verstuurde codewoord precies één bit is veranderd, kunnen we de verstuurde boodschap eenduidig reconstrueren: Het ontvangen woord heeft slechts van een enkel geldig codewoord afstand 1 en van alle andere codewoorden minstens afstand 2 anders zouden er namelijk twee codewoorden met afstand 2 zijn. Door de ontvangen rij van bits naar het eenduidige geldige codewoord met afstand 1 om te zetten, kunnen we dus één fout verbeteren.

> Merk op dat de $(7,4)$-Hamming code heel zuinig is: Als we de woorden die afstand 1 van een geldig codewoord hebben de *buren* van dit codewoord noemen, heeft ieder van de 16 geldige codewoorden 7 buren. Maar omdat verschillende codewoorden minstens afstand 3 hebben, mogen twee codewoorden geen gemeenschappelijke buren hebben. Als we nu de geldige codewoorden en hun buren tellen, zijn dit er $16 + 16 \cdot 7 = 128 = 2^7$ woorden, d.w.z. we hebben alle 7-bit woorden gevonden. In het bijzonder zijn er dus geen woorden van 7 bits die een afstand van minstens 2 van *alle* codewoorden hebben.

Een foutverbeterende code zo als de $(7,4)$-Hamming code wordt als volgt toegepast:

Een boodschap van 4 bits heeft een kans van p^4 om correct ontvangen te worden als we geen redundantie inbouwen.

De $(7,4)$-Hamming code is een 6-uit-7 blok, want we kunnen bij 6 correct ontvangen bits het verstuurde codewoord en dus de originele boodschap eenduidig reconstrueren. De betrouwbaarheid van het versturen met behulp van de $(7,4)$-Hamming code is dus

$$R = p^7 + 7p^6(1-p) = p^7 + 7p^6 - 7p^7 = 7p^6 - 6p^7 = p^4(7p^2 - 6p^3).$$

Dit is een verbetering tegenover het gewone versturen zonder redundantie als $7p^2 - 6p^3 > 1$. Men gaat snel na dat dit het geval is als $p > \frac{1}{2}$.

Als derde variant voor het versturen van 4 bits bekijken we een 3-repetition code. Omdat dit voor ieder bit een 2-uit-3 systeem is, heeft deze vorm van codering de betrouwbaarheid $(3p^2 - 2p^3)^4$.

De tabel hieronder laat voor drie speciale waarden van p zien dat de Hamming-code veel beter is dan het gewone versturen van bits zonder redundantie en bijna zo goed als een 3-repetition code. Merk op dat bij de drievoudige herhaling 12 bits verstuurd worden, terwijl dit er bij de Hamming-code slechts 7 bits zijn, deze is dus veel zuiniger.

	zonder redundantie	Hamming	3-repetition
aantal bits	4	7	12
0.9	0.66	0.85	0.89
0.99	0.9606	0.9980	0.9988
0.999	0.996006	0.999979	0.999988
p	p^4	$7p^6 - 6p^7$	$(3p^2 - 2p^3)^4$

7.2 Levensduur

Vaak is het bij systemen interessant de (verwachte) levensduur te bepalen. Dit is de tijd tot het falen van het systeem. Omdat we de kans dat een systeem normaal werkt met de betrouwbaarheid van het systeem aangeven, kunnen we de levensduur bepalen als we de betrouwbaarheid R als functie $R(t)$ van de tijd opvatten.

Als we met T de (niet bekende) levensduur van een systeem noteren, is de betrouwbaarheid $R(t)$ op het tijdstip t de kans dat T groter is dan t. Als we T als een stochast zien, hebben we dus

$$R(t) = P(T \geq t) = 1 - P(T \leq t).$$

Maar een stochast X met continue kansverdeling wordt meestal door een dichtheidsfunctie $f(x)$ en een verdelingsfunctie $F(x)$ beschreven, waarbij $F(x_0) = P(X \leq x_0) = \int_{-\infty}^{x_0} f(x)\,dx$ geldt.

Als nu $f(t)$ de dichtheidsfunctie voor het falen op tijdstip t is, hebben we $P(T \leq t) = \int_0^t f(\tau)\,d\tau$ en dus

$$R(t) = 1 - \int_0^t f(\tau)\,d\tau.$$

We kunnen de dichtheidsfunctie $f(t)$ ook nog iets anders interpreteren: Volgens de hoofdstelling van de calculus geldt $(\int f(t)\ dt)' = f(t)$, dus hebben we

$$R'(t) = -f(t),$$

de dichtheidsfunctie $f(t)$ geeft dus de snelheid van verandering van de betrouwbaarheid aan. Merk op dat de betrouwbaarheid met de tijd nooit toeneemt, daarom is de afgeleide $R'(t)$ steeds ≤ 0.

Met een trucje uit de calculus kunnen we de verwachtingswaarde $E(T)$ van de levensduur van een systeem rechtstreeks uit de functie $R(t)$ voor de betrouwbaarheid berekenen.

De productregel voor de afgeleide zegt dat

$$(f(x) \cdot g(x))' = f'(x)g(x) + f(x)g'(x).$$

Als we hier op de twee zijden de integraal van nemen en gebruik ervan maken dat $\int (f(x) \cdot g(x))'\ dx = f(x) \cdot g(x)$, volgt hieruit

$$f(x) \cdot g(x) = \int f'(x)g(x)\ dx + \int f(x)g'(x)\ dx.$$

Door een van de integralen naar de andere kant te brengen, krijgen we de regel voor de *partiële integratie*:

$$\int f(x)g'(x)\ dx = f(x) \cdot g(x) - \int f'(x)g(x)\ dx.$$

Voor bepaalde integralen krijgen we

$$\int_a^b f(x)g'(x)\ dx = f(b)g(b) - f(a)g(a) - \int_a^b f'(x)g(x)\ dx.$$

De verwachtingswaarde $E(T)$ is gedefinieerd als

$$E(T) = \int_0^\infty t \cdot f(t)\ dt.$$

Als we hier $f(t)$ door $-R'(t)$ vervangen en de regel van de partiële integratie toepassen, krijgen we

$$E(T) = -\int_0^\infty t \cdot R'(t)\ dt = -t \cdot R(t)\ |_0^\infty + \int_0^\infty R(t)\ dt = \int_0^\infty R(t)\ dt.$$

In de laatste stap moeten we veronderstellen dat de functie $R(t)$ sneller naar 0 gaat dan $\frac{1}{t}$. Dit is bij alle realistische functies voor de betrouwbaarheid het geval, in het bijzonder bij exponentiële verdelingen die we als voorbeelden straks gaan bekijken. Uit deze aanname over $R(t)$ volgt dat $\lim_{t\to\infty} t \cdot R(t) = 0$ en voor $t = 0$ is sowieso $t \cdot R(t) = 0$.

Conclusie: De verwachtingswaarde van de levensduur van een systeem is gelijk aan de integraal over de betrouwbaarheidsfunctie van het systeem.

Tot nu toe hebben we nog geen concrete dichtheidsfunctie $f(t)$ verondersteld. We zullen hier alleen maar één belangrijk speciaal geval voor de verdeling van de levensduur behandelen, namelijk het geval van een exponentieel verdeelde levensduur. Dat dit een belangrijk geval is hebben we in de vorige les bij de Poisson-processen gezien, waar de tussentijden steeds exponentieel verdeeld waren. Als we het falen van componenten als Poisson-proces zien, zijn we dus precies in dit geval. Dit veronderstelt natuurlijk dat het falen van een component onafhankelijk van de geschiedenis van de component is, in het bijzonder vindt geen ouderdomsverzwakking plaats.

Definitie: We zeggen dat de levensduur T van een systeem *exponentieel verdeeld* met parameter λ is, als de dichtheidsfunctie $f(t)$ voor het falen gegeven is door $f(t) = \lambda e^{-\lambda t}$. In dit geval geldt $P(T \leq t) = 1 - e^{-\lambda t}$ en dus

$$R(t) = 1 - (1 - e^{-\lambda t}) = e^{-\lambda t}.$$

De verwachte levensduur van zo'n systeem is

$$E(T) = \int_0^\infty e^{-\lambda t}\, dt = -\frac{1}{\lambda} e^{-\lambda t} \,|_0^\infty = \frac{1}{\lambda}.$$

(We wisten natuurlijk al lang dat een exponentiële verdeling met parameter λ verwachtingswaarde $\frac{1}{\lambda}$ heeft.)

We zullen nu de levensduur voor bepaalde combinaties van componenten met exponentieel verdeelde levensduur bepalen, te weten voor een rijschakeling, een parallelschakeling en een TMR blok.

Rijschakeling

We gaan ervan uit dat we n componenten in een rij hebben waarvan de i-de een exponentieel verdeelde levensduur met parameter λ_i heeft. Voor de betrouwbaarheid van een rijschakeling hadden we gezien dat $R = \prod_{i=1}^n R_i$, dus hebben we nu

$$R(t) = \prod_{i=1}^n R_i(t) = \prod_{i=1}^n e^{-\lambda_i t} = e^{-\lambda_1 t} \cdot \ldots \cdot e^{-\lambda_n t} = e^{-\lambda_1 t - \ldots - \lambda_n t} = e^{-(\sum_{i=1}^n \lambda_i) t}.$$

De levensduur van het systeem is dus exponentieel verdeeld met parameter $\lambda_1 + \ldots \lambda_n = \sum_{i=1}^n \lambda_i$ en de verwachte levensduur is

$$E(T) = \frac{1}{\sum_{i=1}^n \lambda_i}.$$

In het bijzonder is de verwachte levensduur van het systeem korter dan die van elke van zijn componenten. Als alle componenten dezelfde parameter λ hebben, is de verwachte levensduur van het systeem slechts een n-de van de verwachte levensduur van de componenten.

Parallelschakeling

Bij een parallelschakeling is de levensduur van het systeem het maximum van de levensduren van zijn componenten.

De verwachte levensduur van een systeem met twee parallelle componenten met exponentieel verdeelde levensduren met parameters λ_1 en λ_2 vinden we als volgt: De betrouwbaarheid van een gewoon systeem met twee parallelle componenten was $R = 1 - (1 - R_1)(1 - R_2)$, dus hebben we

$$\begin{aligned} R(t) &= 1 - (1 - R_1(t))(1 - R_2(t)) = 1 - (1 - e^{-\lambda_1 t})(1 - e^{-\lambda_2 t}) \\ &= e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}. \end{aligned}$$

We hebben al gezien dat $\int_0^\infty e^{-\lambda t}\ dt = \frac{1}{\lambda}$, daarom geldt voor de verwachte levensduur:

$$\begin{aligned} E(T) &= \int_0^\infty R(t)\ dt = \int_0^\infty e^{-\lambda_1 t}\ dt + \int_0^\infty e^{-\lambda_2 t}\ dt - \int_0^\infty e^{-(\lambda_1+\lambda_2)t}\ dt \\ &= \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1 + \lambda_2}. \end{aligned}$$

De verwachte levensduur is dus korter dan de som van de verwachte levensduren van de componenten. In het geval $\lambda_1 = \lambda_2 = \lambda$ is de verwachte levensduur van het systeem $\frac{3}{2} \cdot \frac{1}{\lambda}$, dus door de parallelle component wordt de levensduur om 50% verhoogd.

Bij een parallelschakeling van n componenten met exponentieel verdeelde levensduren met dezelfde parameter λ laat zich aantonen dat het systeem de verwachte levensduur

$$E(T) = \frac{1}{\lambda}(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n})$$

heeft. Voor grote waarden van n geldt $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} \approx \log(n) + 0.577$, dit betekent dat de levensduur van het systeem voor groeiende n slechts zo als $\log(n)$ toeneemt. Omdat de logaritme heel langzaam groeit is een parallelschakeling dus alleen maar voor 2 of 3 componenten een efficiënte manier om de levensduur te verhogen.

TMR blok

We hadden gezien dat een TMR blok (of 2-uit-3 blok) met betrouwbaarheid R van de componenten betrouwbaarheid $R_{TMR} = 3R^2 - 2R^3$ heeft. Voor de betrouwbaarheid van een TMR blok waarbij de componenten exponentieel verdeelde levensduur met parameter λ hebben, geldt dus

$$R(t) = 3e^{-2\lambda t} - 2e^{-3\lambda t}.$$

Voor de verwachte levensduur $E(T)$ krijgen we

$$\begin{aligned} E(T) &= \int_0^\infty R(t)\ dt = \int_0^\infty 3e^{-2\lambda t}\ dt - \int_0^\infty 2e^{-3\lambda t}\ dt \\ &= -\frac{3}{2\lambda}e^{-2\lambda t}\ |_0^\infty + \frac{2}{3\lambda}e^{-3\lambda t}\ |_0^\infty = \frac{3}{2\lambda} - \frac{2}{3\lambda} = \frac{5}{6} \cdot \frac{1}{\lambda}. \end{aligned}$$

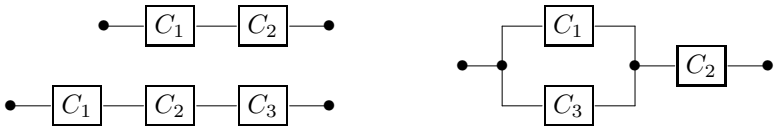
De verwachte levensduur is dus *korter* dan bij de enkele componenten! Dit lijkt paradox, want we hadden gezien dat voor $R > \frac{1}{2}$ de TMR blok een grotere betrouwbaarheid heeft dan R. Maar dit is juist de sleutel voor de verklaring van de paradox. De TMR blok is betrouwbarer dan een van zijn componenten als deze betrouwbaarheid $> \frac{1}{2}$ hebben. Maar de betrouwbaarheid van de componenten is gegeven door $e^{-\lambda t}$ en $e^{-\lambda t} > \frac{1}{2}$ is equivalent met $t < \log(2) \cdot \frac{1}{\lambda} \approx 0.7 \cdot \frac{1}{\lambda}$. De TMR blok is dus alleen maar voor kortere tijdsduren betrouwbarer dan een van de componenten, de grotere betrouwbaarheid van de enkele component bij grotere tijden zorgt ervoor dat de verwachtingswaarde voor de levensduur groter is dan bij de TMR blok.

Belangrijke begrippen in deze les

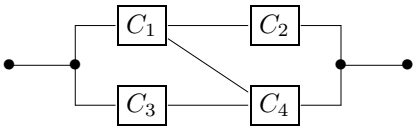
- betrouwbaarheid
- redundantie
- rijschakeling, parallelschakeling, rij-parallel-schakeling
- m-uit-n blok, voter
- foutverbeterende code
- (verwachte) levensduur

Opgaven

47. Laten C_1, C_2 en C_3 drie componenten met bijhorende betrouwbaarheden R_1, R_2 en R_3 zijn. Bereken de betrouwbaarheden van de volgende combinaties van componenten voor algemene R_i en voor de speciale waarden $R_1 = 0.8$, $R_2 = 0.75$, $R_3 = 0.98$.



48. Bereken de betrouwbaarheid van het volgende systeem, afhankelijk van de betrouwbaarheden R_1, R_2, R_3 en R_4 van de enkele componenten.



(Hint: Onderscheid de twee gevallen dat C_1 werkt of niet werkt, die met kans R_1 en $(1 - R_1)$ optreden. Als C_1 niet werkt, blijft van het systeem alleen maar de rijschakeling van C_3 en C_4 over, als C_1 wel werkt is het irrelevant of C_3 werkt, het systeem is dan een rijschakeling van C_1 met de parallelschakeling van C_2 en C_4.)

Stel je hebt twee componenten met een betrouwbaarheid van 90% beschikbaar en twee componenten met een betrouwbaarheid van slechts 80%. Op welke posities moet je de betere componenten plaatsen en welke betrouwbaarheid kun je zo maximaal bereiken?

49. Om een zeker programma uit te voeren, moeten op een multiprocessor-computer met een kans van 95% minstens 2 processoren beschikbaar zijn. Als designer van de multiprocessor-computer kun je verschillende typen van processoren inbouwen: De goedkoopste processor met een betrouwbaarheid van 60% kost 1000€ en elke verhoging van de betrouwbaarheid om 10% kost 800€ extra (dus betaal je 3400€ voor een processor met een betrouwbaarheid van 90%.

    Wat is de goedkopste manier om aan de eisen van het programma te voldoen als je een parallelschakeling van *gelijksoortige* processoren bouwt? Is er een goedkopere manier als je verschillende typen van processoren mag gebruiken?

50. De $(23, 12)$-Golay code is een lineaire code met 2^{12} legale codewoorden van lengte 23 (d.w.z. boodschappen van 12 bits worden door 11 check-bits aangevuld). De legale codewoorden in de Golay code hebben een Hamming-afstand van minstens 7, daarom kan de Golay code 3 fouten verbeteren. We kunnen daarom bij een ontvangen boodschap van 23 bits de oorspronkelijke boodschap eenduidig reconstrueren, als er hoogstens 3 bits zijn veranderd. Stel dat één bit met kans p correct wordt ontvangen. Bereken de betrouwbaarheid van het versturen van een boodschap van 12 bits met behulp van de $(23, 12)$-Golay code.

    Je kunt een boodschap van 12 bits natuurlijk ook als 3 blokken van 4 bits met behulp van de $(7, 4)$-Hamming code versturen. Vergelijk voor $p = 0.9$ en $p = 0.99$ de betrouwbaarheden van het versturen met behulp van de $(23, 12)$-Golay code en met behulp van de $(7, 4)$-Hamming code (in drie blokken). Vergelijk de betrouwbaarheden ook met de betrouwbaarheid bij het versturen met drievoudige herhaling, dus met een 2-uit-3 blok voor elke bit.

51. In een systeem dat twee stappen bevat is elke component verdubbeld. Bepaal voor de volgende twee designs van het systeem de verwachte levensduur als alle componenten een exponentieel verdeelde levensduur met parameter λ hebben. Welke design geeft een grotere levensduur?

    Bepaal de verwachte levensduur van de systemen ook voor het geval dat de componenten C_1 een exponentieel verdeelde levensduur met parameter λ_1 hebben en de componenten C_2 een exponentieel verdeelde levensduur met parameter λ_2.

    Laat zien dat het design in het rechterplaatje voor alle waarden van λ_1 en λ_2 een hogere verwachte levensduur geeft. (Hiervoor is het handig, de verwachte levensduren van de twee designs van elkaar af te trekken en aan te nemen dat $\lambda_2 = 1$ is.)