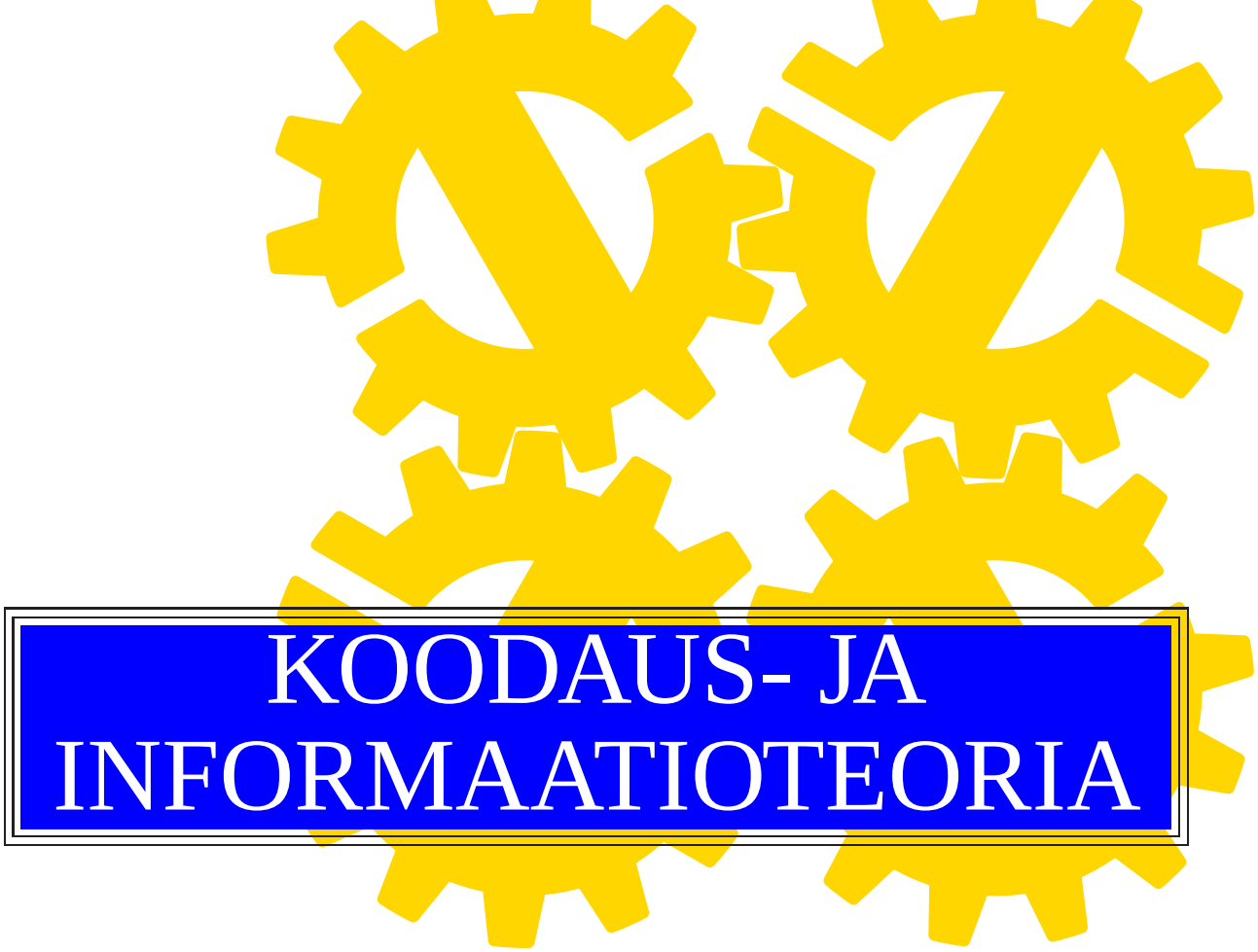


KOODAUS- JA INFORMAATIOTEORIA

Keijo Ruohonen

1999

Sisältöluettelo

Esipuhe

Käsillä oleva moniste muodostaa TTKK:n matematiikan kurssien Koodausteoria ja Informaatioteoria luentorungon. Moniste on tarkoitus täydentää luennoilla esimerkein sekä lisäyksin. Mainitut kurssit on tarkoitettu tukemaan lähinnä signaalien käsittelyyn, tietoliikennetekniikkaan sekä elektroniikkaan liittyviä kursseja. Koska kurssit Koodausteoria ja Informaatioteoria luennoidaan erikseen ja jälkimmäinen ei suinkaan pelkästään ole tarkoitettu tukemaan edellistä, on informaatioteorian osuus kirjoitettu paljolti erilleen koodausteoriasta, toisin kuin monissa kirjoissa. Itseopiskelijalle suositellaan monisteen ohella käytettäväksi joitain runsaasti esimerkkejä sisältäviä kirjoja (esim. ANDERSON & MOHAN, SWEENEY ja YAGLOM & YAGLOM).

Keijo Ruohonen

I LUKU
LINEAARISET KOODIT

1. Lohkokoodit

Viestiä lähetettäessä on käytössä tietyt symbolit, ns. *viestisymbolit.* Viesti jaetaan tasapituisiin osiin, ns. *lohkoihin* eli *viestisanoihin.* Koodattaessa kukin viestin lohko korvataan ns. *koodisymboleista* muodostetulla *koodisanalla*. Kahta eri lohkoa ei saa vastata sama koodisana, mikä takaa koodatun viestin dekoodautuvuuden. Usein koodisymbolit ovat samat kuin viestisymbolit. Jos kaikki koodisanat ovat samanpituisia, on kyseessä ns. *lohkokoodi.* Yhteinen koodisanojen pituus on ns. *koodin pituus*. (Koodausteoria käyttää nimenomaan lohkokoodeja.) Kaikki koodisanat muodostavat joukkona *koodin.* Jos koodisymboleja on kaksi (yleensä bitit 0 ja 1) puhutaan *binäärikoodista*.

Koodauksen tarkoitus on saada aikaan mahdollisuus dekoodattaessa paljastaa ja korjata tietty määrä tiedonsiirrossa tai -talletuksessa syntyviä virheitä koodisanaa kohti.

Jatkossa merkitään viesti- ja koodilohkoja lihavalla: Esimerkiksi

$$\mathbf{m} = m_1 m_2 \cdots m_k$$

on viestisana, jonka symbolit ovat $m_1, m_2, \ldots, m_k$ ja jonka pituus on k. Viestisanojen pituutta merkitään jatkossa useimmiten k:lla ja koodin pituutta n:llä. Luku n – k on ns. *redundanssi.*

ESIMERKKI. Yksinkertainen tapa saada aikaan virheiden paljastamis/korjaamiskyky on käyttää toistokoodausta. Tällöin viestisanaa **m** vastaava koodisana on

$$\mathbf{c} = \mathbf{m}^r = \underbrace{m_1 m_2 \cdots m_k \cdots m_1 m_2 \cdots m_k}_{\text{r kertaa}}.$$

Tällainen koodaus pystyy joko paljastamaan r – 1 virhettä tai korjaamaan $\lfloor (r-1)/2 \rfloor$ virhettä. Jos siis r = 2, voidaan yhden virheen esiintyminen havaita, mutta sitä ei pystytä korjaamaan.

Kahden koodisanan $\mathbf{c}_1$ ja $\mathbf{c}_2$ ns. (*Hamming-)etäisyys* $d(\mathbf{c}_1,\mathbf{c}_2)$ on niissä olevien erojen lukumäärä. Koodin C ns. *minimietäisyys,* merkitään $d_{min}(C)$, on pienin esiintyvä kahden eri koodisanan etäisyys.

ESIMERKKI. Jos binäärikoodin koodisymbolit ovat bitit 0 ja 1, niin koodisanojen $\mathbf{c}_1$ = 001001100 ja $\mathbf{c}_2$ = 101101100 etäisyys on $d(\mathbf{c}_1,\mathbf{c}_2) = 2$. Koodin pituus on tässä 9.

ESIMERKKI. Valitaan toistokoodauksessa viestisymboleiksi 0 ja 1, viestisanan pituudeksi 3 ja vielä r = 3. Saadun binäärikoodin C pituus on 9, koodisanoja on $2^3 = 8$ kpl ja $d_{min}(C) = 3$.

Koodia on tapana ajatella “geometrisesti” pistejoukkona kaikkien mahdollisten koodisymboleista muodostettujen n:n (= koodin pituus) pituisten sanojen muodostamassa “avaruudessa”. Etäisyys d on geometrisen vastineensa kaltainen sikälikin, että se noudattaa *kolmioepäyhtälöä,* ts.

$$d(\mathbf{c}_1,\mathbf{c}_2) \le d(\mathbf{c}_1,\mathbf{c}_3) + d(\mathbf{c}_3,\mathbf{c}_2)$$

(helppo todeta). Jos $\mathbf{c}$ on koodisana, niin $\mathbf{c}$-keskinen R-säteinen *pallo* on

$$B(\mathbf{c},R) = \{\mathbf{d} \mid d(\mathbf{c},\mathbf{d}) \le R\}.$$

Seuraava perustulos on aivan yksinkertaisesti pääteltävissä:

LAUSE 1. **(i)** Koodi C korjaa t virhettä tarkalleen silloin, kun $d_{min}(C) \ge 2t + 1$ (eli silloin, kun koodisanakeskiset t-säteiset pallot eivät leikkaa toisiaan).

(ii) Koodi C paljastaa s virhettä tarkalleen silloin, kun $d_{min}(C) \ge s + 1$ (eli silloin, kun mikään koodisana ei ole toisen koodisanan s-säteisessä pallossa). ■

HUOM! Jos koodin C halutaan samanaikaisesti korjaavan t virhettä ja paljastavan s virhettä, tarvitaan toisenlainen ehto. Tietysti $s \ge t$. (Eo. Esimerkissä oleva 9-pituinen toistokoodi ei ilmeisestikään pysty yhtaikaiseen yhden virheen korjaamiseen ja kahden virheen paljastukseen.) Geometrisesti esitettynä ehto on seuraava: Jokaiselle koodisanalle $\mathbf{c} \in C$ jokaisen toisen C:n koodisanan etäisyys pallon $B(\mathbf{c},s)$ sanoista on suurempi kuin t. Kolmioepäyhtälöä käyttäen voi todeta, että tämä on sama kuin ehto $d_{min}(C) \ge t + s + 1$ (jätetään lukijalle).

Koodi C on *täydellinen* t virhettä korjaava koodi, jos

(1) $d_{min}(C) \geq 2t + 1$ (ks. Lause **1i**) ja

(2) $\bigcup_{\mathbf{c} \in C} B(\mathbf{c}, t)$ käsittää kaikki koodiaakkoston sanat, ts. jokainen koodiaakkoston sana kuuluu johonkin koodisanakeskiseen t-säteiseen palloon.

Täydellinen (t virhettä korjaava) koodi toimii “ekonomisesti”, mutta se korjaa väärin tilanteet, joissa lohkossa on virheitä enemmän kuin t kpl.

Pyyhkiymä on virhe, jossa symbolin paikalle ei tule toinen symboli, vaan symboli katoaa kokonaan. Näin ollen pyyhkiymien kohdat ovat välittömästi saatavissa, mutta eivät pois pyyhkiytyneet symbolit. Aivan ilmeisesti koodi C korjaa p pyyhkiymää tarkalleen silloin, kun

$$d_{min}(C) \geq p + 1.$$

Yleisemmin koodi C korjaa p pyyhkiymää ja t “tavallista” virhettä tarkalleen silloin, kun

$$d_{min}(C) \geq 2t + p + 1.$$

2. Ekskursio: Alkukunnat

Lineaarisen koodin koodisanat käsitetään symboliensa muodostamiksi vektoreiksi (jotka vielä muodostavat lineaarisen aliavaruuden). Jotta tällainen olisi mahdollista, pitää koodisymboleille määritellä “tutut” laskutoimitukset +, –, × ja / siten, että tavalliset laskusäännöt pitävät paikkansa. Jos joukossa $\mathbb{F}$ on määritelty yhteenlasku + ja kertolasku × sekä nolla-alkio $\overline{0}$ ja ykkösalkio $\overline{1}$ siten, että

(1) + ja × ovat liitännäiset sekä vaihdannaiset,

(2) osittelulait pätevät,

(3) jokaisella alkiolla a on vasta-alkio –a, ts. $a + (-a) = \overline{0}$,

(4) jokaisella muulla alkiolla a paitsi $\overline{0}$:lla on käänteisalkio a^{-1}, ts. $a \times a^{-1} = \overline{1}$, ja

(5) $\overline{0}$ ja $\overline{1}$ “käyttäytyvät” kuten pitää, ts. $a + \overline{0} = a$ ja $a \times \overline{1} = a$, (ja lisäksi vielä $\overline{1} \neq \overline{0}$),

niin näillä laskuoperaatioilla varustettuna $\mathbb{F}$ on ns. *kunta*. Tuttuja kuntia ovat reaalilukujen kunta $\mathbb{R}$, rationaalilukujen kunta $\mathbb{Q}$ ja kompleksilukujen

kunta $\mathbb{C}$. Nämä ovat kuitenkin äärettömiä eivätkä tule kysymykseen koodisymbolien joukkoina.

Koodisymbolien joukoksi sopivan kunnan pitääkin olla joukkona äärellinen, ns. *äärellinen kunta* eli *Galois'n kunta*. Yksinkertaisia äärellisiä kuntia ovat ns. alkukunnat eli jäännösluokkien modulo alkuluku p muodostamat kunnat (p = 2,3,5,7,...).

Merkitään $\mathbb{Z}_m$:llä jäännösluokkia modulo m. $\mathbb{Z}_m$:n alkiot ovat

$$\begin{aligned}
\overline{0} &= \{\, km \mid k = 0,\pm1,\pm2,\ldots \,\},\\
\overline{1} &= \{\, km + 1 \mid k = 0,\pm1,\pm2,\ldots \,\},\\
\overline{2} &= \{\, km + 2 \mid k = 0,\pm1,\pm2,\ldots \,\},\\
&\ldots\\
\overline{m-1} &= \{\, (k+1)m - 1 \mid k = 0,\pm1,\pm2,\ldots \,\}.
\end{aligned}$$

Jäännösluokkaa $\bar{i}$ voidaan merkitä ja se voidaan antaa minkä tahansa edustajansa avulla, ts. $\bar{i} = \overline{km+i}$. Erityisesti $\overline{0} = \overline{m}$ ja $-\overline{1} = \overline{m-1}$. Laskuoperaatiot määritellään nyt seuraavasti:

$$\bar{i} + \bar{j} = \overline{i+j} \quad \text{ja} \quad \bar{i} \times \bar{j} = \overline{ij}.$$

On helppo todeta, että operaatiot on näin hyvin määritelty eli tulos ei riipu siitä mitkä edustajat i ja j jäännösluokista valitaan. Nolla-alkio on luonnollisesti $\overline{0}$ ja ykkösalkio $\overline{1}$, edelleen $-\bar{i} = \overline{-i}$. Yleisesti $\overline{0}$:n lisäksi $\mathbb{Z}_m$:ssä voi olla muitakin alkioita, joilla ei ole käänteisalkiota. Jos m = p on alkuluku, niin muilla kuin $\overline{0}$:lla on kuitenkin aina käänteisalkio (ks. kurssi 73265 Symbolinen analyysi 1). Näin saadut kunnat $\mathbb{Z}_p$ eli GF(p) ovat ns. *alkukunnat.*

ESIMERKKI. Yksinkertaisin alkukunta on *binäärikunta* $\mathbb{Z}_2$ eli GF(2), jonka alkioita $\overline{0}$ ja $\overline{1}$ voidaan pitää bitteinä 0 ja 1. Laskuoperaatiot saadaan tauluista

+	0	1
0	0	1
1	1	0

ja

×	0	1
0	0	0
1	0	1

Kaikki "tavalliset" lineaarialgebran käsitteet kuten vektorit ja matriisit sekä näiden laskuoperaatiot*, determinantit, ortogonaalisuus, lineaarinen riippumattomuus ja riippuvuus, dimensio, kanta, vektorijoukon virittämä aliavaruus ja sen ortogonaalinen komplementti, matriisin rangi ym. voidaan nyt määritellä vektoreille, joiden alkiot ovat jonkin alkukunnan al-

---

* Koodausteoriassa vektorit kirjoitetaan yleensä vaakavektoreiksi $\mathbf{v} = (v_1,\ldots,v_n)$ ja vektori kerrotaan matriisilla oikealta.

kioita. (Huomaa kuitenkin, ettei vektorien pistetulolla varsinaisesti ole mitään tekemistä vektorien välisen “kulman” kanssa, eikä ominaisarvoja ja -vektoreita yleisesti tarvita.) Koska nämä käsitteet ovat tuttuja peruskursseilta reaalilukukunnan tapauksessa ja ovat alkukuntien tapauksessa hyvin samankaltaisia, ei niitä tässä sen tarkemmin määritellä. Kaikkien (alku)kunnan $\mathbb{F}$ alkioista muodostettujen n-ulotteisten vektorien joukkoa merkitään $\mathbb{F}^n$:llä.

HUOM! On muitakin äärellisiä kuntia kuin alkukunnat ja näitäkin tarvitaan tehokkaissa koodausmenetelmissä. Asiaan palataan myöhemmin **III.1**:ssä.

3. Lineaariset koodit

Kun koodisymbolien joukoksi valitaan alkukunta $\mathbb{F}$ (tai yleisesti äärellinen kunta), voidaan sanat ja vektorit täydellisesti samaistaa:

$$\mathbf{c} = c_1c_2\cdots c_n = (c_1,c_2,\ldots,c_n).$$

Koodia $C \subseteq \mathbb{F}^n$ sanotaan *lineaariseksi koodiksi,* jos se on $\mathbb{F}^n$:n lineaarinen aliavaruus, ts.

(i) jos $\mathbf{c}_1$ ja $\mathbf{c}_2$ ovat koodisanoja, niin samoin on $\mathbf{c}_1 + \mathbf{c}_2$ (vektorien yhteenlasku) ja

(ii) jos $\mathbf{c}$ on koodisana ja f on kunnan $\mathbb{F}$ alkio, niin $f\mathbf{c}$ on koodisana.

Näin ollen nollavektori $\mathbf{0}$ on aina lineaarisen koodin koodisana ja samoin on kunkin koodisanan $\mathbf{c}$ vastavektori $-\mathbf{c}$. (Huomaa, että alkukunnille $\mathbb{F}$ ehto **(ii)** seuraa **(i)**:stä ja on siis itse asiassa tarpeeton, sillä jos i on positiivinen kokonaisluku, niin

$$\bar{\imath}\mathbf{c} = \underbrace{\mathbf{c}+\cdots+\mathbf{c}}_{\text{i kpl}}.)$$

Lineaarista koodia, jonka pituus on n ja dimensio k (lineaarisena aliavaruutena), sanotaan *(n,k)-koodiksi.*

Vektorin $\mathbf{v}$ *Hamming-pituus* eli *Hamming-paino* $w(\mathbf{v})$ on siinä olevien nollaalkiosta $\bar{0}$ eroavien komponenttien lukumäärä. Näin ollen

$$d(\mathbf{v}_1,\mathbf{v}_2) = w(\mathbf{v}_1 - \mathbf{v}_2) \quad \text{ja} \quad w(\mathbf{v}) = d(\mathbf{v},\mathbf{0})$$

ja lineaarisen koodin C minimietäisyys $d_{min}(C)$ on pienin positiivinen koodisanan paino.

ESIMERKKI. $\mathbb{F}$:n alkioita symboleina käyttäen muodostetut toistokoodit ovat lineaarisia.

Koska lineaarinen koodi C on lineaarinen aliavaruus, sillä on kanta

$$\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_k$$

(itse asiassa tietysti useampiakin kantoja, mutta ei äärettömän monta). Koodisanan $\mathbf{c}$ valinta tarkoittaakin sen kantaesityksessä tarvittavien lineaariyhdelmän $\mathbf{c} = d_1\mathbf{g}_1 + d_2\mathbf{g}_2 + \cdots + d_k\mathbf{g}_k$ kerrointen $d_1,d_2,\ldots,d_k$ valintaa. Koodattaessa pitää siis kutakin viestisanaa $\mathbf{m}$ kohti määrittää vastaavan koodisanan kantaesityksen kertoimet. Yksinkertaisesti tämä käy, mikäli valitaan viestisymbolien joukoksi myös $\mathbb{F}$ ja kertoimiksi viestisymbolit itse. Jos siis viestisana on

$$\mathbf{m} = m_1m_2\cdots m_k = (m_1,m_2,\ldots,m_k),$$

niin koodauksessa tulee sitä vastaamaan koodisana

$$\mathbf{c} = m_1\mathbf{g}_1 + m_2\mathbf{g}_2 + \cdots + m_k\mathbf{g}_k.$$

Huomaa, että viestisanoja on nyt tarkalleen yhtä paljon kuin koodisanoja, ts. q^k kpl, missä q on kunnan $\mathbb{F}$ alkioiden lukumäärä. Valitsemalla eri kantoja saadaan erilaisia koodauksia.

Viestisanat muodostavat viestiavaruuden $\mathbb{F}^k$ ja koodaus voidaan käsittää lineaarikuvaukseksi $\mathbb{F}^k$:lta C:lle. Tällainen lineaarikuvaus voidaan suorittaa matriisia, ns. *generoijamatriisia,* käyttäen:

$$\mathbf{c} = \mathbf{mG},$$

missä

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{pmatrix}.$$

Matriisin $\mathbf{G}$ (oikea) nolla-avaruus muodostuu niistä vektoreista $\mathbf{h}$, joille $\mathbf{G}\mathbf{h}^T = \mathbf{0}_k^T$. Tunnetusti nolla-avaruus on $\mathbb{F}^n$:n lineaarinen aliavaruus, ja sen dimensio on $n - k$. Näin ollen nolla-avaruudessa on q^{n-k} vektoria. Valitaan nolla-avaruuden kanta $\mathbf{h}_1,\mathbf{h}_2,\ldots,\mathbf{h}_{n-k}$ ja muodostetaan niistä matriisi

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_{n-k} \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k,1} & h_{n-k,2} & \cdots & h_{n-k,n} \end{pmatrix},$$

ns. *tarkistusmatriisi*. (Valitusta kannasta riippuen mahdollisia tarkistusmatriiseja on useita.) Silloin

$$\mathbf{G}\mathbf{H}^{\mathrm{T}} = \mathbf{O}_{k\times(n-k)} \quad \text{ja} \quad \mathbf{H}\mathbf{G}^{\mathrm{T}} = \mathbf{O}_{(n-k)\times k}.$$

$\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_k$ puolestaan muodostavat näin **H**:n nolla-avaruuden kannan ja mainittu nolla-avaruus on siis C . Näin ollen koodisanat ovat tarkalleen ne sanat **c**, jotka toteuttavat ns. *tarkistusyhtälön*

$$\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0}_{n-k}.$$

Tämä onnistuu Maple-ohjelmistollakin (ensin annetaan latauskäsky with(linalg)):

```
> G:=array([[1,0,1,1,0],[0,1,1,0,1],[1,1,1,1,1],[0,0,0,0,0],[0,0,0,0,0]]);
```

$$G := \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> Nullspace(G) mod 2;
```

$$\{[1, 0, 0, 1, 0], [0, 1, 0, 0, 1]\}$$

Dekoodaus olisi helppoa, jos pitäisi vain löytää kutakin koodisanaa vastaava viestisana, mutta dekoodauksessa pitää myös korjata tai paljastaa haluttu määrä syntyneitä virheitä. Jos saatu (mahdollisesti) virheellinen sana on **r** ja vastaava koodisana **c**, niin $\mathbf{e} = \mathbf{r} - \mathbf{c}$ on *virhesana* eli *virhevektori*. Virheiden lukumäärä on virhesanan paino w(**e**). Dekoodauksen tarkoitus on löytää virhesana, jolloin oikea koodisana **c** saadaan välittömästi **r**:stä: $\mathbf{c} = \mathbf{r} - \mathbf{e}$. Binäärikoodin tapauksessa riittää itse asiassa löytää vain virhekohdat.

Dekoodauksessa on usein käyttökelpoinen saadun sanan $\mathbf{r} = \mathbf{c} + \mathbf{e}$ (koodisana + virhesana), ns. *syndromi*

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = (\mathbf{c} + \mathbf{e})\mathbf{H}^{\mathrm{T}} = \mathbf{e}\mathbf{H}^{\mathrm{T}}.$$

Koodisanan syndromi on aina **0** ja vastaanotetun sanan **r** syndromi on sama kuin virheen **e** syndromi. Näin ollen syndromi sisältää koodisanasta **c** riippumatonta tietoa virheestä **e**.

APULAUSE. Jos jonkin koodisanan paino on d, niin tarkistusmatriisissa **H** on d lineaarisesti riippuvaa saraketta.

Todistus. Valitaan koodisana $\mathbf{c} = (c_1,\ldots,c_n)$, jonka paino on d. Näin ollen **c**:ssä on tarkalleen d nolla-alkiosta eroavaa komponenttia $c_{i_1},\ldots,c_{i_d}$. Koska **c** on koodisana, on $\mathbf{c}\mathbf{H}^{\mathrm{T}} = \mathbf{0}_{n-k}$. ■

LAUSE 2. Lineaarisen koodin C minimietäisyys on vähintään d tarkalleen silloin, kun mitkä tahansa d – 1 tarkistusmatriisin **H** saraketta ovat lineaarisesti riippumattomat.

Todistus. Jos $d_{min}(C) \geq d$, niin mitkään d – 1 tarkistusmatriisin **H** eri saraketta $\mathbf{f}_{i_1}^T,\ldots,\mathbf{f}_{i_{d-1}}^T$ eivät voi olla lineaarisesti riippuvat. Muussa tapauksessa olisi sellaiset kertoimet $c_{i_1},\ldots,c_{i_{d-1}}$, jotka kaikki eivät ole nolla-alkioita, että $c_{i_1}\mathbf{f}_{i_1} + \cdots + c_{i_{d-1}}\mathbf{f}_{i_{d-1}} = \mathbf{0}_{n-k}$ ja näin ollen sellainen koodisana $\mathbf{c} \neq \mathbf{0}$, että $w(\mathbf{c}) \leq d - 1$. Jos taas mitkä tahansa d – 1 tarkistusmatriisin **H** saraketta ovat lineaarisesti riippumattomat, niin Apulauseen nojalla $d_{min}(C) \geq d$. ■

4. Systemaattinen koodaus

Jos generoijamatriisi on muotoa $\mathbf{G} = (\mathbf{I}_k \,|\, \mathbf{P})$, missä $\mathbf{I}_k$ on k×k-identiteettimatriisi ja **P** on k×(n – k)-matriisi, kyseessä on ns. *systemaattinen koodaus*. Systemaattisessa koodauksessa k ensimmäistä symbolia muodostavat viestisanan ja loput symbolit ovat *tarkistussymboleja*. (Usein tosin toisin päin, ks. **II.4**.)

HUOM! Binäärikunnan $\mathbb{Z}_2$ tapauksessa tarkistus on ns. *pariteetintarkistus* ja puhutaan *pariteetintarkistusmatriisista, pariteetintarkistussymboleista*, jne..

LAUSE 3. Jos lineaarisen koodin generoijamatriisi on $\mathbf{G} = (\mathbf{I}_k \,|\, \mathbf{P})$ (systemaattinen koodaus), niin $\mathbf{H} = \left(-\mathbf{P}^T \,\middle|\, \mathbf{I}_{n-k}\right)$ on vastaava tarkistusmatriisi. Vastaavasti, jos lineaarisen koodin tarkistusmatriisi on $\mathbf{H} = (\mathbf{Q} \,|\, \mathbf{I}_{n-k})$, niin $\mathbf{G} = \left(\mathbf{I}_k \,\middle|\, -\mathbf{Q}^T\right)$ on vastaava (systemaattinen) generoijamatriisi.

Todistus. Näytetään vain ensimmäinen toteamus (toinen menee vastaavasti). **H** on (n – k)×n-matriisi, jonka vaakarivit ovat lineaarisesti riippumattomat. Edelleen

$$\mathbf{G}\mathbf{H}^T = (\mathbf{I}_k \,|\, \mathbf{P})\left(\frac{-\mathbf{P}}{\mathbf{I}_{n-k}}\right) = -\mathbf{P} + \mathbf{P} = \mathbf{O}_{k\times(n-k)}. \quad \blacksquare$$

ESIMERKKI. Toistokoodin (ks. Esimerkki s. 1) generoijamatriisi on

$$\mathbf{G} = \underbrace{(\mathbf{I}_k \,|\, \mathbf{I}_k \,|\cdots|\, \mathbf{I}_k)}_{\text{r kpl}}.$$

Kyseessä on systemaattinen koodaus. Lauseen **2** mukaan vastaava tarkistusmatriisi on

$$\mathbf{H} = \left(\begin{array}{c|c} -\mathbf{I}_k & \\ \hline -\mathbf{I}_k & \\ \hline \vdots & \mathbf{I}_{n-k} \\ \hline -\mathbf{I}_k & \end{array} \right).$$

(Koska n = rk, niin n – k = (r – 1)k.)

5. Standardidekoodauskaavio

Joukkoja

$$\mathbf{e} + C = \{ \mathbf{e} + \mathbf{c} \mid \mathbf{c} \in C \},$$

missä **e** on mielivaltainen virhesana, kutsutaan *sivuluokiksi.* Sivuluokka $\mathbf{e} + C$ sisältää ne sanat, joiksi virhe **e** muuntaa koodisanat. Erityisesti $C = \mathbf{0} + C$.

APULAUSE. Jos kahden sivuluokan leikkaus ei ole tyhjä, niin ne ovat samat.

Todistus. Jos $(\mathbf{e}_1 + C) \cap (\mathbf{e}_2 + C)$ ei ole tyhjä, niin on olemassa sana

$$\mathbf{e}_1 + \mathbf{c}_1 = \mathbf{e}_2 + \mathbf{c}_2,$$

missä $\mathbf{c}_1, \mathbf{c}_2 \in C$. Näin ollen $\mathbf{e}_1 \in \mathbf{e}_2 + C$ ja $\mathbf{e}_2 \in \mathbf{e}_1 + C$, joten sivuluokat sisältyvät toisiinsa ja ovat siis samat. ■

Jokainen sana $\mathbf{v} \in \mathbb{F}^n$ kuuluu johonkin sivuluokkaan ja sivuluokat ovat samankokoiset. Näin ollen sivuluokassa on q^k sanaa ja sivuluokkia on q^{n-k} kpl (eli sama määrä kuin generoijamatriisin nolla-avaruuden alkioita). (q on $\mathbb{F}$:n alkioluku.)

Standardidekoodauskaavio muodostetaan seuraavalla tavalla:

(1) Järjestetään koodisanat jonoksi aloittaen **0**:sta:

$$\mathbf{c}_1 = \mathbf{0}, \mathbf{c}_2, \mathbf{c}_3, \ldots, \mathbf{c}_{q^k}.$$

Näin saadaan kaavion ensimmäinen rivi.

(2) Etsitään kustakin sivuluokasta painoltaan pienin alkio (mahdollisuuksia voi olla useita). Näin saadaan ns. *sivuluokkien johtajat*

$$\mathbf{e}_1 = \mathbf{0}, \mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_{q^{n-k}}.$$

Näistä muodostetaan kaavion ensimmäinen sarake. (C :n johtaja on tietysti $\mathbf{0}$.)

(3) Kaavion i:nnen rivin ja j:nnen sarakkeen alkio on $\mathbf{e}_i + \mathbf{c}_j$.

ESIMERKKI. Binäärikoodin C (kaavion ylin rivi) standardidekoodauskaavio on

johtaja	sivuluokka						
00000	10010	01010	00101	11000	10111	01111	11101
10000	00010	11010	10101	01000	00111	11111	01101
00001	10011	01011	00100	11001	10110	01110	11100
10001	00011	11011	10100	01001	00110	11110	01100

Standardidekoodauskaaviota käytetään dekoodauksessa seuraavasti: Etsitään kaaviosta saatu sana $\mathbf{r}$ ja dekoodataan se vastaavan sarakkeen ylimmäksi sanaksi $\mathbf{c} \in C$. Tulos on luonnollisesti vastaava oikea viestisana vain siinä tapauksessa, että virhe $\mathbf{e}$ oli $\mathbf{r}$:n sisältävän sivuluokan johtaja. Menettely on näin käyttökelpoinen lähinnä vain binäärikoodien tapauksessa ja silloin, kun sivuluokkien johtajiksi on saatu “edustava” kokoelma virheitä.

Täydellinen kaavio sisältää q^{n-k} riviä ja q^k saraketta, mikä on yleensä liian paljon. Syndromeja käyttäen saadaan tilantarvetta pienennettyä. Sivuluokan alkioilla on nimittäin sama syndromi ja eri sivuluokkien alkioilla eri syndromit, kuten helposti voi todeta. Koska saadusta sanasta $\mathbf{r}$ on helposti laskettavissa sen syndromi $\mathbf{s} = \mathbf{r}\mathbf{H}^T$, riittää muodostaa syndromien luettelo ja liittää kuhunkin syndromiin vastaavan sivuluokan johtaja. Dekoodattaessa lasketaan ensin saadun sanan $\mathbf{r}$ syndromi $\mathbf{s}$ ja etsitään luettelosta vastaava sivuluokan johtaja $\mathbf{e}$. $\mathbf{r}$ dekoodataan sitten $\mathbf{r} - \mathbf{e}$:ksi. Tämäkään menettely ei ole kovin käyttökelpoinen, jos $n - k$ on suuri.

6. Hamming-koodit

Tarkistusmatriisi määrittelee lineaarisen koodin siinä kun generoijamatriisikin. Lauseen **2** nojalla lineaarisen koodin C minimietäisyys on vähintään 3 tarkalleen silloin, kun sen tarkistusmatriisin $\mathbf{H}$ mitkään kaksi saraketta eivät ole lineaarisesti riippuvat. Lineaarinen koodi on *Hamming-koodi*, jos sen tarkistusmatriisi sisältää suurimman mahdollisen määrän parittain lineaarisesti riippumattomia sarakkeita eli suurimman mahdollisen määrän sarakkeita, joita ei saada toisistaan vakiolla (eli kunnan $\mathbb{F}$ alkiolla) kertomalla.

Paljonko sitten Hamming-koodin tarkistusmatriisissa on sarakkeita? Ilmeisesti voidaan olettaa, että kunkin sarakkeen ylhäältälukien ensimmäinen nolla-alkiosta eroava alkio on ykkösalkio. Sarakkeita, joissa ylin alkio on ykkösalkio, on selvästi q^{n-k-1} kpl (q on kunnan $\mathbb{F}$ alkioluku). Edelleen sarakkeita, joiden ylin alkio on nolla-alkio ja toiseksi ylin alkio ykkösalkio, on q^{n-k-2} kpl, jne.. Kaiken kaikkiaan sarakkeita on siis

$$q^{n-k-1} + q^{n-k-2} + \ldots + q + 1 = \frac{q^{n-k} - 1}{q - 1} = n$$

kpl. Merkitään $n - k = r$. Silloin Hamming-koodin pituus ja viestisanan pituus ovat

$$n = \frac{q^r - 1}{q - 1} \quad \text{ja} \quad k = \frac{q^r - 1}{q - 1} - r.$$

Samalla tuli esille eräs tapa muodostaa Hamming-koodin tarkistusmatriisi **H**, joka ei ole sen huonompi kuin muutkaan. Voidaankin olettaa, että **H** saadaan juuri tällä tavalla. Vastaavan generoijamatriisin saa vaikkapa käyttämällä Lausetta **3**. Huomaa, että **H**:ssa on mukana kaikki r sellaista saraketta, joissa yksi alkioista on ykkösalkio ja muut nolla-alkioita, ts. koodaus on olennaisesti systemaattinen. Näin ollen **H** kannattaa kirjoittaa muotoon $\mathbf{H} = \left(\mathbf{Q}^T \mid \mathbf{I}_r\right)$. Triviaalien tapausten poissulkemiseksi oletetaan, että $n \geq 3$.

Kuten todettiin, Lauseesta **2** seuraa, että Hamming-koodin minimietäisyys on ainakin 3. Itse asiassa se on tarkalleen 3, sillä yo. tavalla saatu **H** sisältää aina lineaarisesti riippuvat kolme saraketta

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

LAUSE 4. Hamming-koodi on täydellinen yhden virheen korjaava koodi.

Todistus. 1-säteiset koodisanakeskiset pallot ovat erilliset ja niitä on q^k kpl. Kussakin tällaisessa pallossa on ilmeisesti $n(q - 1) + 1$ alkiota. Siispä mainituissa palloissa on yhteensä $q^k(n(q - 1) + 1) = q^n$ alkiota eli ne peittävät koko $\mathbb{F}^n$:n. ■

Yhden virheen korjaava dekoodaus Hamming-koodilla on helppoa. Jos saadussa sanassa $\mathbf{r} = \mathbf{c} + \mathbf{e}$ on yksi virhe, ts.

$$\mathbf{e} = a\mathbf{e}_i,$$

missä $\mathbf{e}_i$ on vektori, jonka i:s alkio on ykkösalkio ja muut nolla-alkioita, niin syndromi on

$$\mathbf{s} = \mathbf{e}\mathbf{H}^T = a\mathbf{e}_i\mathbf{H}^T = a\mathbf{f}_i,$$

missä $\mathbf{f}_i^T$ on **H**:n i:s sarake. Koska **H**:n sarakkeet ovat erilaisia, paljastuu näin syndromista virheen paikka i ja virhe a (= ensimmäinen **s**:n nolla-alkiosta eroava komponentti). Dekoodaus tuottaa oikean koodisanan

$$\mathbf{c} = \mathbf{r} - a\mathbf{e}_i.$$

Jos taas syndromi on $\mathbf{0}$, ei virhettä ole tapahtunut eikä mitään korjausta tarvita.

ESIMERKKI. Binäärisen Hamming-koodin pituus on $n = 2^r - 1$ ja viestisanan pituus $k = 2^r - r - 1$. Näin ollen binäärinen Hamming-koodi on $(2^r - 1, 2^r - r - 1)$-koodi. Pariteetintarkistusmatriisin $\mathbf{H}$ sarakkeina esiintyvät kaikki $2^r - 1$ kpl r:n pituisia bittivektoreita paitsi nollavektori. Jos nämä kirjoitetaan $\mathbf{H}$:ssa sellaiseen järjestykseen, että i:nnen sarakkeen bitit muodostavat luvun i binääriesityksen, on dekoodaus todella helppoa: Syndromi nimittäin on virheen paikan binääriesitys! (Itse virhe on ilmeinen, sillä virheen sattuessa 0 vaihtuu 1:ksi tai 1 vaihtuu 0:ksi.) Valitettavasti vain $\mathbf{H}$ ei tällöin ole muotoa $\left(\mathbf{Q}^T \mid \mathbf{I}_r\right)$ ja generoijamatriisin etsiminen ei mene aivan suoraan Lausetta **3** käyttäen. Systemaattinen koodaus saadaan kuitenkin myös pariteetintarkistusyhtälöä käyttäen. Yhtälöstä $\mathbf{c}\mathbf{H}^T = \mathbf{0}_r$ voidaan nimittäin ratkaista esimerkiksi $c_{k+1},\ldots,c_n$ lausuttuna ensimmäisten koodisymbolien $c_1,\ldots,c_k$ avulla. Näin voidaan $c_1c_2\cdots c_k$ valita viestisanaksi $\mathbf{m}$ ja loput koodisymbolit ovat pariteetintarkistussymboleja. Esimerkiksi valitaan $r = 3$, jolloin saadaan (7,4)-Hamming-koodi ja pariteetintarkistusmatriisi on

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Ratkaistaan yhtälö $\mathbf{c}\mathbf{H}^T = \mathbf{0}_3$ Maplella (näin pieni menisi toki käsipelissäkin):

```
> solve({c4+c5+c6+c7=0,c2+c3+c6+c7=0,c1+c3+c5+c7=0},{c5,c6,c7}):
> " mod 2;
                    {c6= c4+ c1+ c3, c7= c2+ c4+ c1, c5= c3+ c2+ c4}
```

Systemaattisessa koodauksessa valitaan $c_1c_2c_3c_7 = \mathbf{m}$ ja loput koodisymbolit lasketaan näistä yhtälöistä. (Menettelyä voi käyttää myös muille kuin binäärisille Hamming-koodeille.) Binäärisiä Hamming-koodeja ja niiden variantteja käytetään paljon tiedon talletuksessa, ks. RAO & FUJIWARA.

II LUKU
SYKLISET KOODIT

1. Syklinen koodi

Lineaarinen koodi on *syklinen,* jos aina kun $c_1 \cdots c_{n-1} c_n$ on koodisana, niin samoin on $c_n c_1 \cdots c_{n-1}$. Syklisen koodin koodisanan $c_1 \cdots c_{n-1} c_n$ kaikki sykliset siirrot $c_{n-i} \cdots c_n c_1 \cdots c_{n-i-1}$ (i = 0,...,n – 2) ovat näin ollen myös koodisanoja.

Syklinen koodi on luonnollinen valinta, mikäli koodausta ajatellaan operaationa, joka toteutetaan lineaarisen järjestelmän avulla. Koodisana

$$\mathbf{c} = c_1 c_2 \cdots c_n$$

samaistetaan paitsi vektoriin $(c_1,c_2,\ldots,c_n)$ myös sen (äärelliseen) z-muunnokseen

$$\mathbf{c}(z) = c_1 + c_2 z + \cdots + c_n z^{n-1}.$$

$\mathbf{c}(z)$ on formaalinen* enintään astetta n – 1 oleva z:n polynomi, ns. *koodipolynomi,* jonka kertoimet ovat kunnan $\mathbb{F}$ alkioita. Koodaus "taajuustasolla" tapahtuu kertomalla viestin (äärellinen) z-muunnos $\mathbf{m}(z)$ "siirtofunktiolla" $\mathbf{g}(z)$, joka sekin on formaalinen enintään astetta n – 1 oleva z:n polynomi, ns. *generoijapolynomi.* Jottei asteluku nousisi yli sallitun n – 1:n, pitää tulosta vielä redusoida korvaamalla z^n jokaisessa esiintymässään $\overline{1}$:llä, ts. merkitsemällä $z^n - \overline{1} = \overline{0}$. Asia on parhaiten esitettävissä polynomialgebraa käyttäen.

2. Ekskursio: Polynomirenkaat ja äärellinen z-muunnos

Algebrallisena käsitteenä rengas määritellään samaan tyyliin kuin kunta (ks. **I.2**), mutta jättäen pois joitain vaatimuksia. Jos joukossa $\mathbb{M}$ on määritelty yhteenlasku + ja kertolasku × sekä nolla-alkio $\overline{0}$ ja ykkösalkio $\overline{1}$ siten, että

---

* Formaalinen polynomi ei ole sama kuin sen antama polynomikuvaus. Esimerkiksi polynomin $1 + z + z^2$ antama $\mathbb{Z}_2$:n kuvaus on vakiofunktio 1.

(1) + ja × ovat liitännäiset sekä vaihdannaiset,

(2) osittelulait pätevät,

(3) jokaisella alkiolla a on vasta-alkio –a, ts. a + (–a) = $\overline{0}$, ja

(4) $\overline{0}$ ja $\overline{1}$ "käyttäytyvät" kuten pitää, ts. a + $\overline{0}$ = a ja a × $\overline{1}$ = a,

niin näillä laskuoperaatioilla varustettuna $\mathbb{M}$ on ns. *rengas*.

HUOM! Tarkemmin sanoen tällainen $\mathbb{M}$ on ns. vaihdannainen 1-rengas, varsinainen rengas on algebrallisesti vieläkin yleisempi käsite. Ks. kurssi 73115 Algebra 1. Jatkossa renkaasta puhuttaessa kyseessä on siis aina juuri tällainen vaihdannainen 1-rengas.

Jokainen kunta on myös rengas. Tuttuja renkaita, jotka eivät ole kuntia, ovat mm. kokonaislukujen rengas $\mathbb{Z}$ sekä erilaiset polynomirenkaat, esimerkiksi rationaali-, reaali-, kompleksi- ja kokonaiskertoimisten polynomien renkaat $\mathbb{Q}$[z], $\mathbb{R}$[z], $\mathbb{C}$[z] ja $\mathbb{Z}$[z]. Myös $\mathbb{Z}_m$ eli jäännösluokat modulo m muodostavat renkaan.

Polynomit, jotka muodostetaan formaalisesti käyttäen kunnan $\mathbb{F}$ alkioita kertoimina, muodostavat ns. $\mathbb{F}$:n *polynomirenkaan* $\mathbb{F}$[z]. Koodausteoriassa tarvitaan nimenomaan äärellisten kuntien polynomirenkaita. Tavalliseen tapaan $\mathbb{F}$[z]:n nollapolynomi samaistetaan $\mathbb{F}$:n nolla-alkioon $\overline{0}$ ja vakiopolynomit vastaaviin $\mathbb{F}$:n alkioihin. Polynomin **p**(z) aste deg(**p**(z)) määritellään edelleen tavalliseen tapaan korkeimman esiintyvän z:n potenssin eksponentiksi. Nollapolynomin asteeksi voidaan sopia vaikkapa –1.

Kunnan polynomirenkaan polynomeille **a**(z) ja **b**(z) ≠ $\overline{0}$ on määritelty yksikäsitteinen osamäärä **q**(z) ja jakojäännös **j**(z), ts. voidaan kirjoittaa

$$\mathbf{a}(z) = \mathbf{q}(z)\mathbf{b}(z) + \mathbf{j}(z), \qquad \deg(\mathbf{j}(z)) < \deg(\mathbf{b}(z)).$$

(Käytännössä tällainen jakolasku voidaan suorittaa "pitkänä jakolaskuna" paperilla, myös esimerkiksi Maple-ohjelmisto osaa jakaa polynomeja äärellisissä kunnissa.) **j**(z):n sanotaan olevan **a**(z):n *jäännös modulo* **b**(z), merkitään $\overline{\mathbf{a}(z)}$, vrt. vastaava käsite kokonaisluvuille **I.2**:ssa. **b**(z) on ns. *moduli*. Kaikki jäännökset modulo **b**(z) muodostavat ns. *jäännösluokkarenkaan* eli *tekijärenkaan* $\mathbb{F}[z]/\langle \mathbf{b}(z) \rangle$. On helppo nähdä, samaan tapaan kuin kokonaisluvuille, että jäännökset modulo **b**(z) voidaan voidaan antaa ja niillä voidaan laskea "edustajan välityksellä", ts.

$$\overline{\mathbf{a}_1(z)} + \overline{\mathbf{a}_2(z)} = \overline{\mathbf{a}_1(z) + \mathbf{a}_2(z)} \quad \text{ja} \quad \overline{\mathbf{a}_1(z)} \times \overline{\mathbf{a}_2(z)} = \overline{\mathbf{a}_1(z)\mathbf{a}_2(z)}.$$

(Jakojäännöstä **j**(z) vastaavan jäännösluokan edustajia ovat ne polynomit **a**(z), joiden jäännös modulo **b**(z) on **j**(z). Jäännösluokka voidaan myös sa-

maistaa edustajiensa muodostamaan joukkoon.) Näin ollen $\mathbb{F}[z]/\langle \mathbf{b}(z)\rangle$ on todellakin rengas.

Vektorin $\mathbf{v} = (v_1,v_2,\ldots,v_n) \in \mathbb{F}^n$ *äärellinen z-muunnos* on $\mathbb{F}[z]$:n polynomi

$$\mathbf{v}(z) = v_1 + v_2 z + \cdots + v_n z^{n-1},$$

mutta samaistettuna jäännösluokkaan modulo $\mathbf{S}(z) = z^n - \overline{1}$. Itse asiassa äärelliset z-muunnokset ovat siis jäännösluokat modulo $z^n - \overline{1}$ ja niillä lasketaan redusoiden aina z^n ykkösalkioksi, sillä $z^n - \overline{1} = \overline{0}$ eli $z^n = \overline{1}$.

3. Syklinen koodi polynomi-ihanteena

Ellei sekaannusta satu, ts. $\deg(\mathbf{v}(z)) < n$, samaistetaan polynomi $\mathbf{v}(z)$, sen edustama jäännösluokka $\overline{\mathbf{v}(z)}$, vektori $\mathbf{v} = (v_1,v_2,\ldots,v_n)$ ja sana $\mathbf{v} = v_1v_2\cdots v_n$ täysin. Koodisanan $\mathbf{c}(z) = c_1 + \cdots + c_{n-1}z^{n-2} + c_n z^{n-1}$ syklinen siirto tapahtuu silloin z:lla eli koodisanalla $\overline{0}\,\overline{1}\,\overline{0}\cdots\overline{0}$ kertomalla:

$$\overline{z\mathbf{c}(z)} = c_n + c_1 z + \cdots + c_{n-1}z^{n-1}.$$

Joukkoa

$$\langle \mathbf{f}(z)\rangle = \{\ \overline{\mathbf{v}(z)\mathbf{f}(z)}\ |\ \mathbf{v}(z) \in \mathbb{F}[z]/\langle \mathbf{S}(z)\rangle\ \}$$

kutsutaan $\mathbf{f}(z)$:n *generoimaksi ihanteeksi* eli *ideaaliksi.* Huomaa. että tällainen ihanne on kaikkien niiden tulosten joukko, jotka saadaan operoimalla "siirtofunktiolla" $\mathbf{f}(z)$ äärellisiin z-muunnoksiin $\mathbf{v}(z)$. Aivan ilmeisesti koodina $\langle \mathbf{f}(z)\rangle$ on syklinen koodi, mutta $\mathbf{f}(z)$ ei (välttämättä) ole sen generoijapolynomi.

LAUSE 5. Koodi on syklinen n-pituinen koodi täsmälleen silloin, kun se on muotoa

$$\{\ \mathbf{v}(z)\mathbf{g}(z)\ |\ \mathbf{v}(z) \in \mathbb{F}[z]/\langle \mathbf{S}(z)\rangle,\ \deg(\mathbf{v}(z)) < n - \deg(\mathbf{g}(z))\ \} = \langle \mathbf{g}(z)\rangle,$$

missä $\mathbf{g}(z)$ on $\mathbf{S}(z)$:n (eli $z^n - \overline{1}$:n) tekijä, ns. *generoijapolynomi.*

Todistus. Todetaan ensin, että

$$\{\ \mathbf{v}(z)\mathbf{g}(z)\ |\ \mathbf{v}(z) \in \mathbb{F}[z]/\langle \mathbf{S}(z)\rangle,\ \deg(\mathbf{v}(z)) < n - \deg(\mathbf{g}(z))\ \}$$

todellakin on tällöin $\mathbf{g}(z)$:n generoima ihanne $\langle \mathbf{g}(z)\rangle$ ja näin ollen syklinen koodi. Aivan ilmeisesti se sisältyy mainittuun ihanteeseen $\langle \mathbf{g}(z)\rangle$. Jos $\mathbf{g}(z)$ on $\mathbf{S}(z)$:n tekijä, voidaan kirjoittaa $\mathbf{S}(z) = \mathbf{g}(z)\mathbf{h}(z)$ ja $\deg(\mathbf{h}(z)) = n -$

$\deg(\mathbf{g}(z))$. Valitaan mielivaltainen $\mathbf{v}(z) \in \mathbb{F}[z]/\langle \mathbf{S}(z) \rangle$, tulkitaan se $\mathbb{F}[z]$:n polynomiksi ja jaetaan $\mathbf{h}(z)$:lla:

$$\mathbf{v}(z) = \mathbf{q}(z)\mathbf{h}(z) + \mathbf{j}(z), \quad \deg(\mathbf{j}(z)) < n - \deg(\mathbf{g}(z)).$$

Modulo $\mathbf{S}(z)$ on $\mathbf{v}(z)\mathbf{g}(z)$ näin ollen sama kuin $\mathbf{j}(z)\mathbf{g}(z)$ ja $\mathbf{v}(z)$:n asteen voidaan siis olettaa olevan pienempi kuin $n - \deg(\mathbf{g}(z))$.

Oletetaan sitten, että C on syklinen koodi ja valitaan sen asteeltaan pienin nollapolynomista eroava koodipolynomi $\mathbf{g}(z)$. Syklisyydestä ja lineaarisuudesta johtuen myös polynomit $\mathbf{v}(z)\mathbf{g}(z)$, $\mathbf{v}(z) \in \mathbb{F}[z]/\langle \mathbf{S}(z) \rangle$, ovat koodipolynomeja, sillä jos $\mathbf{v}(z) = v_1 + v_2 z + \cdots + v_n z^{n-1}$, niin

$$\mathbf{v}(z)\mathbf{g}(z) = v_1\mathbf{g}(z) + v_2 z\mathbf{g}(z) + \cdots + v_n z^{n-1}\mathbf{g}(z).$$

Jaetaan $\mathbf{S}(z)$ $\mathbf{g}(z)$:llä:

$$\mathbf{S}(z) = \mathbf{q}(z)\mathbf{g}(z) + \mathbf{j}(z), \quad \deg(\mathbf{j}(z)) < \deg(\mathbf{g}(z)).$$

Modulo $\mathbf{S}(z)$ on $\mathbf{j}(z)$ näin ollen sama kuin $-\mathbf{q}(z)\mathbf{g}(z)$ ja on siis koodipolynomi. $\mathbf{g}(z)$:n asteen minimaalisuudesta johtuen pitää $\mathbf{j}(z)$:n olla nollapolynomi, jolloin $\mathbf{g}(z)$ on $\mathbf{S}(z)$:n tekijä. Samaan tapaan voidaan jakolaskulla todeta, että $\mathbf{g}(z)$ on jokaisen C:n koodipolynomin tekijä. C on näin ollen esitettyä muotoa. ■

Syklisen koodin* koodaus voidaan suorittaa generoijapolynomilla $\mathbf{g}(z)$ kertomalla, kunhan viestisana $\mathbf{m} = m_1 m_2 \cdots m_k$ samaistetaan viestipolynomiksi $\mathbf{m}(z) = m_1 + m_2 z + \cdots + m_k z^{k-1}$, missä $k = n - \deg(\mathbf{g}(z))$:

$$\mathbf{c}(z) = \mathbf{m}(z)\mathbf{g}(z).$$

ESIMERKKI. Lineaarinen toistokoodi (ks. Esimerkki s. 1) on syklinen ja sen koodaus voidaan suorittaa generoijapolynomilla $\overline{1} + z^k + z^{2k} + \cdots + z^{(r-1)k}$.

Koska syklisen koodin generoijapolynomi $\mathbf{g}(z)$ on $\mathbf{S}(z)$:n tekijä, voidaan kirjoittaa

$$\mathbf{S}(z) = \mathbf{g}(z)\mathbf{h}(z),$$

missä $\mathbf{h}(z)$ on ns. *tarkistuspolynomi.* Nimi on sattuva (vrt. tarkistusmatriisi), sillä modulo $\mathbf{S}(z)$ on $\mathbf{g}(z)\mathbf{h}(z)$ näin $\overline{0}$. Edelleen saadaan

---

* Muidenkin koodien kuin syklisten koodien koodaus voidaan suorittaa tällä tavoin, mutta koodipolynomit eivät tällöin muodosta lineaarisen järjestelmän koko tulosjoukkoa ja koodaus on tietyssä mielessä tehotonta. Esimerkiksi (lineaarisen) koodin {000,101} koodaus voidaan suorittaa polynomilla $1 + z^2$ (joka ei ole $1 + z^3$:n tekijä).

LAUSE 6. Jos syklisen koodin tarkistuspolynomi on $\mathbf{h}(z)$, niin $\mathbf{c}(z)$ on koodipolynomi täsmälleen silloin, kun $\mathbf{c}(z)\mathbf{h}(z)$ on nollapolynomi modulo $\mathbf{S}(z)$.

Todistus. Jos $\mathbf{c}(z)$ on koodipolynomi, niin se on muotoa $\mathbf{m}(z)\mathbf{g}(z)$, missä $\mathbf{g}(z)$ on tarkistuspolynomia $\mathbf{h}(z)$ vastaava generoijapolynomi. Näin ollen tällöin $\mathbf{c}(z)\mathbf{h}(z) = \mathbf{m}(z)\mathbf{g}(z)\mathbf{h}(z) = \mathbf{m}(z)\mathbf{S}(z)$ on nollapolynomi modulo $\mathbf{S}(z)$.

Oletetaan sitten, että $\mathbf{c}(z)\mathbf{h}(z)$ on nollapolynomi modulo $\mathbf{S}(z)$. Jaetaan $\mathbf{c}(z)$ generoijapolynomilla $\mathbf{g}(z)$:

$$\mathbf{c}(z) = \mathbf{q}(z)\mathbf{g}(z) + \mathbf{j}(z), \qquad \deg(\mathbf{j}(z)) < \deg(\mathbf{g}(z)).$$

Silloin myös $\mathbf{j}(z)\mathbf{h}(z)$ on nollapolynomi modulo $\mathbf{S}(z)$, ts. muotoa $\mathbf{p}(z)\mathbf{S}(z)$. Mutta astelukuja ajatellen tämä on mahdollista vain, jos $\mathbf{j}(z)$ on itse nollapolynomi, jolloin $\mathbf{c}(z) = \mathbf{q}(z)\mathbf{g}(z)$ on koodisana. ■

Yleensä generoijapolynomi $\mathbf{g}(z)$ valitaan *pääpolynomiksi*, ts. siinä valitaan z^{n-k}:n kerroin ykkösalkioksi. Tämä ei mitenkään rajoita tilannetta. Seurauksena myös vastaava tarkistuspolynomi on pääpolynomi.

Analogisesti aikaisemmin esillä olleen syndromin määritelmän mukaisesti polynomia

$$\overline{\mathbf{r}(z)\mathbf{h}(z)} = \mathbf{s}(z)$$

kutsutaan saadun sanan $\mathbf{r}(z)$ *syndromipolynomiksi.* Syndromipolynomi määräytyy vain virhesanasta eikä riipu koodisanasta. Jos $\mathbf{s}(z) \neq \overline{0}$, tiedetään virheen sattuneen. Vaihtoehtoisesti virhe voitaisiin paljastaa jakamalla $\mathbf{r}(z)$ $\mathbf{g}(z)$:lla, jolloin jakojäännös on syndromipolynomi. Jos jakojäännös ei ole nollapolynomi, tiedetään virheen tapahtuneen.

Jos syklisen koodin C_1 generoijapolynomi $\mathbf{g}_1(z)$ on toisen syklisen koodin C_2 generoijapolynomin $\mathbf{g}_2(z)$ tekijä, niin C_2 sisältyy C_1:een. Syklistä koodia sanotaankin *maksimaaliseksi*, jos sen generoijapolynomi $\mathbf{g}(z)$ on jaoton, ts. (vakiopolynomien lisäksi) $\mathbf{g}(z)$:llä ei ole muita tekijöitä kuin se itse.

ESIMERKKI. Valitaan $n = 7$ ja $\mathbb{F} = \mathbb{Z}_2$. Polynomin $z^7 - 1$ eli $z^7 + 1$ jaottomat tekijät ovat $1 + z$, $1 + z + z^3$ ja $1 + z^2 + z^3$, kuten kokeilemalla voi havaita. Nämä kukin generoivat maksimaalisen syklisen koodin. Koodissa $\langle 1 + z \rangle$ on 2^6 koodisanaa, $\langle 1 + z + z^3 \rangle$:ssa $2^4 = 16$ koodisanaa ja samoin koodissa $\langle 1 + z^2 + z^3 \rangle$. Kaikki mahdolliset sykliset 7-pituiset binäärikoodit saadaan etsimällä vielä muut generoijapolynomit:

$$(1 + z)(1 + z + z^3) = 1 + z^2 + z^3 + z^4 \qquad ((7,3)\text{-koodi}),$$

$$(1 + z)(1 + z^2 + z^3) = 1 + z + z^2 + z^4 \qquad ((7,3)\text{-koodi}),$$

$$(1 + z + z^3)(1 + z^2 + z^3) = 1 + z + z^2 + z^3 + z^4 + z^5 + z^6 \qquad ((7,1)\text{-koodi}).$$

Viimemainittu (7,1)-koodi on itse asiassa toistokoodi $\{\mathbf{0},1111111\}$. Lisäksi voidaan ottaa mukaan "degeneroituneet tapauksetkin" eli valita generoijapolynomeiksi

$$1 \qquad ((7,7)\text{-koodi } \mathbb{F}^7),$$

$$1 + z^7 \qquad ((7,0)\text{-koodi } \{\mathbf{0}\}).$$

Kaikkiaan näitä koodeja on siis $2^3 = 8$ kpl.

Jos $z^n - \overline{1}$:n jako jaottomiin pääpolynomeihin $\mathbf{p}_1(z),\ldots,\mathbf{p}_m(z)$ on

$$z^n - \overline{1} = \mathbf{p}_1(z)^{k_1}\cdots\mathbf{p}_m(z)^{k_m},$$

niin syklisiä n-pituisia koodeja on $(k_1 + 1)\cdots(k_m + 1)$ kpl, kun mukaan luetaan myös "degeneroituneet" koodit $\mathbb{F}^n$ ja $\{\mathbf{0}\}$ (vrt. yo. Esimerkki).

Syklisen koodin koodaus on yleensä helposti digitaalisesti toteutettavissa (onhan se lineaarinen järjestelmä), samoin syndromin lasku. Ks. ANDERSON & MOHAN.

4. Syklisen koodin systemaattinen koodaus

Edellisen pykälän koodausmenettely ei ole systemaattinen. Systemaattinen koodaus saadaan kuitenkin vähän muuntaen koodausmenettelyä. Kerrotaan viestipolynomi $\mathbf{m}(z)$ ennen koodausta z^{n-k}:lla ja jaetaan se sitten generoijapolynomilla $\mathbf{g}(z)$:

$$z^{n-k}\mathbf{m}(z) = \mathbf{q}(z)\mathbf{g}(z) + \mathbf{u}(z), \quad \deg(\mathbf{u}(z)) < n - k.$$

Silloin $-\mathbf{u}(z) + z^{n-k}\mathbf{m}(z) = \mathbf{q}(z)\mathbf{g}(z)$ on koodipolynomi ja vastaava koodivektori on $(-\mathbf{u},\mathbf{m})$. Viestisana $\mathbf{m}$ onkin tässä koodisanan lopussa (eikä alussa kuten **I.4**:ssä käsitellyssä systemaattisessa koodauksessa) ja alkuosa $-\mathbf{u}$ muodostuu tarkistussymboleista. Koska polynomeja $-\mathbf{u}(z) + z^{n-k}\mathbf{m}(z)$ on aivan yhtä monta (q^k kpl) kuin koodipolynomeja yhteensä, saadaan näin yleinen systemaattinen koodausmenettely.

5. Syklisen koodin generoija- ja tarkistusmatriisit

Viestipolynomin $\mathbf{m}(z)$ kertomista generoijapolynomilla

$$\mathbf{g}(z) = g_1 + g_2 z + \cdots + g_{n-k+1}z^{n-k}$$

vastaa vektorin $\mathbf{m}$ kertominen matriisilla

$$\mathbf{G} = \begin{pmatrix} g_1 & g_2 & g_3 & \cdots & g_k & \cdots & g_{n-k+1} & \overline{0} & \overline{0} & \cdots & \overline{0} \\ \overline{0} & g_1 & g_2 & \cdots & g_{k-1} & \cdots & g_{n-k} & g_{n-k+1} & \overline{0} & \cdots & \overline{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{0} & \overline{0} & \overline{0} & \cdots & g_1 & \cdots & g_{n-2k+2} & g_{n-2k+3} & g_{n-2k+4} & \cdots & g_{n-k+1} \end{pmatrix}.$$

Näin ollen tämä matriisi **G** voidaan ottaa generoijamatriisiksi. (Ilmeisesti **G**:n rivit ovat lineaarisesti riippumattomat.)

Jos taas vastaava tarkistuspolynomi on

$$\mathbf{h}(z) = h_1 + h_2 z + \cdots + h_{k+1} z^k,$$

niin siitä saadaan matriisi

$$\mathbf{H} = \begin{pmatrix} \overline{0} & \overline{0} & \overline{0} & \cdots & \overline{0} & \cdots & \overline{0} & h_{k+1} & h_k & \cdots & h_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{0} & h_{k+1} & h_k & \cdots & h_2 & \cdots & \overline{0} & \overline{0} & \overline{0} & \cdots & \overline{0} \\ h_{k+1} & h_k & h_{k-1} & \cdots & h_1 & \cdots & \overline{0} & \overline{0} & \overline{0} & \cdots & \overline{0} \end{pmatrix}.$$

Nyt polynomitulossa $\mathbf{g}(z)\mathbf{h}(z)$ z^i:n kerroin on

$$\sum_{j=0}^{i} g_{i+1-j} h_{j+1},$$

kun sovitaan, että

$$g_{n-k+2} = g_{n-k+3} = \cdots = \overline{0},$$

$$g_0 = g_{-1} = \cdots = \overline{0},$$

$$h_{k+2} = h_{k+3} = \cdots = \overline{0},$$

$$h_0 = h_{-1} = \cdots = \overline{0},$$

kuten matriiseissa on. Mutta $\mathbf{g}(z)\mathbf{h}(z) = z^n - \overline{1}$, joten

$$\sum_{j=0}^{i} g_{i+1-j} h_{j+1} = \overline{0} \quad (i = 1, \ldots, n-1).$$

Nämä ovat juuri ne summat, jotka esiintyvät muodostettaessa matriisitulo $\mathbf{G}\mathbf{H}^T$, joten se on $= \mathbf{O}_{k\times(n-k)}$. Koska **H**:n rivit ilmeisestikin ovat lineaarisesti riippumattomat, on **H** tarkistusmatriisi, joka vastaa generoijamatriisia **G**. Jos edelleen kyseessä on edellisen pykälän systemaattinen koodaus, ovat koodisanat $-\mathbf{u}_i(z) + z^{n-k+i}$ $(i = 0,\ldots,k-1)$ lineaarisesti riippumattomat, joten niistä voidaan muodostaa koodin kanta. Kun merkitään

$$\mathbf{u}_i(z) = u_{i1} + u_{i2} z + \cdots + u_{i,n-k} z^{n-k-1} \qquad (i = 0,\ldots,k-1),$$

saadaan näin generoijamatriisi

$$\mathbf{G} = \left(\begin{array}{ccc|ccc} -u_{01} & \cdots & -u_{0,n-k} & \overline{1} & \overline{0} & \cdots & \overline{0} \\ -u_{11} & \cdots & -u_{1,n-k} & \overline{0} & \overline{1} & \cdots & \overline{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -u_{k-1,1} & \cdots & -u_{k-1,n-k} & \overline{0} & \overline{0} & \cdots & \overline{1} \end{array}\right) = \left(-\mathbf{U} \,|\, \mathbf{I}_k\right).$$

Käyttäen Lausetta **3** vähän muunnetussa muodossa saadaan edelleen vastaavaksi tarkistusmatriisiksi

$$\mathbf{H} = \left(\mathbf{I}_{n-k} \,|\, \mathbf{U}^T\right).$$

6. Meggitt-dekoodaus

Jos syndromi

$$\mathbf{s}(z) = \overline{\mathbf{r}(z)\mathbf{h}(z)} = \overline{\left(\left(\mathbf{c}(z) + \mathbf{e}(z)\right)\mathbf{h}(z)\right)} = \overline{\mathbf{e}(z)\mathbf{h}(z)}$$

ei ole nollapolynomi, tiedetään virheen tapahtuneen. Koska

$$\overline{z^i \mathbf{s}(z)} = \overline{\left(z^i \mathbf{e}(z)\right)\mathbf{h}(z)},$$

voidaan kokeilemalla läpi arvot i = 0,...,n – 1 siirtyä tilanteeseen, jossa virhe tapahtuu viimeisessä koodisymbolissa, mikä helpottaa dekoodausta. Tähän periaatteeseen perustuvaa dekoodausta kutsutaan *Meggitt-dekoodaukseksi.* Pseudokoodi

```
BEGIN INPUT s(z)
IF  s(z) = 0 THEN  OUTPUT  "Ei havaittuja virheitä"
ELSE
      BEGIN
      i := n – 1;
      WHILE s(z) ≠ s_1(z),...,s_{q – 1}(z) AND i > 0 DO
            BEGIN
            i := i – 1;
            s(z) := z·s(z)‾;
            END;
      IF s(z) = s_j(z) THEN  OUTPUT "Virhe a_j paikassa i + 1"
      ELSE  OUTPUT "Virhe paljastui, ei voida korjata"
      END;
END.
```

missä $\mathbf{s}_1(z),\ldots,\mathbf{s}_{q-1}(z)$ ovat virheitä $a_1 z^{n-1},\ldots,a_{q-1}z^{n-1}$ vastaavat syndromit, antaa yhden virheen korjaavan Meggitt-dekoodauksen. Meggitt-de-

koodauksella voidaan korjata myös useampia virheitä käyttäen syndromitaulua, missä ovat edustettuina kaikkien niiden korjattavissa olevien virheiden syndromit, joissa yksi virheistä tapahtuu viimeisessä koodisymbolissa. Myös jakojäännössyndromia voitaisiin käyttää Meggitt-dekoodauksessa. Meggitt-dekoodaus on myös varsin helposti digitaalisesti toteutettavissa (Ks. ANDERSON & MOHAN.)

7. Golay-koodit

Binäärinen Golay-koodi on binäärinen syklinen (23,12)-koodi. Jaetaan $z^{23} + 1$ jaottomiin tekijöihinsä Maplella:

```
> Factor(z^23+1) mod 2;
```

$$(z^{11} + z^9 + z^7 + z^6 + z^5 + z + 1)(z + 1)(z^{11} + z^{10} + z^6 + z^5 + z^4 + z^2 + 1)$$

$1 + z$:n generoima syklinen koodi ei ole kovin kiinnostava. Sen sijaan tekijät $1 + z + z^5 + z^6 + z^7 + z^9 + z^{11}$ ja $1 + z^2 + z^4 + z^5 + z^6 + z^{10} + z^{11}$ kumpikin generoivat binäärisen Golay-koodin. Tämän koodin minimietäisyys on 7 ja se on täydellinen kolmen virheen korjaava koodi (vaikea todistaa, ks. esimerkiksi BLAHUT, tietokoneella asia voidaan tietysti todentaa).

Ternäärinen Golay-koodi on syklinen (11,6)-koodi, jonka koodiaakkosto on $\mathbb{Z}_3$. Jaetaan $z^{11} - \overline{1}$ jaottomiin tekijöihinsä Maplella:

```
> Factor(z^11-1) mod 3;
```

$$(z^5 + z^4 + 2z^3 + z^2 + 2)(z + 2)(z^5 + 2z^3 + z^2 + 2z + 2)$$

Jälleen tekijät $\overline{2} + \overline{2}z + z^2 + \overline{2}z^3 + z^5$ sekä $\overline{2} + z^2 + \overline{2}z^3 + z^4 + z^5$ kumpikin generoivat ternäärisen Golay-koodin. Tämän koodin minimietäisyys on 5 ja se on täydellinen kahden virheen korjaava koodi.

HUOM! Hamming-koodien ja Golay-koodien lisäksi muut täydelliset lineaariset koodit ovatkin sitten yksinkertaisia toistokoodeja. Tämän oudon ja erittäin syvällisen tuloksen todisti lopullisesti v. 1971 suomalainen matemaatikko (TTKK:ssakin aikanaan matematiikan professorina ollut) Aimo Tietäväinen.

III LUKU
BCH–KOODIT

1. Ekskursio: Äärelliset kunnat

Alkukunnat saatiin **I.2**:ssa muodossa $\mathbb{Z}_p$ eli jäännösluokkina modulo jokin alkuluku p. Valitaan nyt alkukunnan $\mathbb{Z}_p$ (eli GF(p)) polynomirenkaan $\mathbb{Z}_p[z]$ polynomi **p**(z), joka on jaoton pääpolynomi (ks. **II.2–3**). Jäännökset modulo **p**(z) muodostavat jäännösluokkarenkaan $\mathbb{Z}_p[z]/\langle \mathbf{p}(z)\rangle$, ks. **II.2**. Aivan samalla tavoin kuin tehtiin osoitettaessa, että jokaisella muulla $\mathbb{Z}_p$:n alkiolla kuin nolla-alkiolla on käänteisalkio, voidaan näyttää, että jokaisella muulla $\mathbb{Z}_p[z]/\langle \mathbf{p}(z)\rangle$:n alkiolla kuin nolla-alkiolla on käänteisalkio, ks. kurssi 73265 Symbolinen analyysi 1. Tätä varten tarvitaan $\mathbb{Z}_p[z]$:n polynomien suurin yhteinen tekijä sekä Eukleideen algoritmi.

$\mathbb{Z}_p[z]$:n polynomien **a**(z) ja **b**(z) *suurin yhteinen tekijä* on korkeinta astetta oleva oleva polynomi **d**(z), joka jakaa tasan sekä **a**(z):n että **b**(z):n, merkitään **d**(z) = syt(**a**(z),**b**(z)). Huomaa, että tällainen suurin yhteinen tekijä ei ole yksikäsitteinen, sillä jos **d**(z) = syt(**a**(z),**b**(z)) ja $\bar{\imath} \in \mathbb{Z}_p - \{\bar{0}\}$, niin myös $\bar{\imath}$**d**(z) on syt(**a**(z),**b**(z)). (Usein vaaditaankin lisäksi, että **d**(z) on pääpolynomi.)

Jos **d**(z) = syt(**a**(z),**b**(z)), niin, kuten nähdään, on sellaiset polynomit $\mathbf{c}_1(z)$ ja $\mathbf{c}_2(z)$, että

$$\mathbf{d}(z) = \mathbf{c}_1(z)\mathbf{a}(z) + \mathbf{c}_2(z)\mathbf{b}(z).$$

Merkitään SYT(**a**(z),**b**(z)) = (**d**(z),$\mathbf{c}_1(z)$,$\mathbf{c}_2(z)$) ja oletetaan, että deg(**a**(z)) ≤ deg(**b**(z)). *Eukleideen algoritmi* on seuraava rekursio:

(1) Jos **a**(z) = $\bar{0}$, niin tulostetaan SYT(**a**(z),**b**(z)) = (**b**(z), $\bar{0}$, $\bar{1}$) ja lopetetaan.

(2) Jos **a**(z) ≠ $\bar{0}$, niin etsitään **b**(z):n jäännös **j**(z) modulo **a**(z), ts. kirjoitetaan **b**(z) = **q**(z)**a**(z) + **j**(z), missä deg(**j**(z)) < deg(**a**(z)). Etsitään sitten SYT(**j**(z),**a**(z)) = (**d**(z),$\mathbf{e}_1(z)$,$\mathbf{e}_2(z)$). Koska **d**(z) = $\mathbf{e}_1(z)$**j**(z) + $\mathbf{e}_2(z)$**a**(z), on nyt **d**(z) = syt(**a**(z),**b**(z)) ja

$$\mathbf{d}(z) = \mathbf{e}_1(z)\mathbf{j}(z) + \mathbf{e}_2(z)\mathbf{a}(z) = \mathbf{e}_1(z)\big(\mathbf{b}(z) - \mathbf{a}(z)\mathbf{q}(z)\big) + \mathbf{e}_2(z)\mathbf{a}(z)$$

$$= \big(\mathbf{e}_2(z) - \mathbf{e}_1(z)\mathbf{q}(z)\big)\mathbf{a}(z) + \mathbf{e}_1(z)\mathbf{b}(z).$$

Tulostetaan $\mathrm{SYT}\big(\mathbf{a}(z),\mathbf{b}(z)\big) = \big(\mathbf{d}(z),\mathbf{e}_2(z) - \mathbf{e}_1(z)\mathbf{q}(z),\mathbf{e}_1(z)\big)$ ja lopetetaan.

Rekursio on päättyvä, koska

$$\min\big(\deg(\mathbf{j}(z)),\deg(\mathbf{a}(z))\big) < \min\big(\deg(\mathbf{a}(z),\deg(\mathbf{b}(z))\big),$$

ts. aina kutsuttaessa SYT ko. minimiarvo pienenee.

Jos nyt $\mathbf{p}(z)$ on jaoton ja $\mathbf{a}(z)$ ei ole muotoa $\mathbf{c}(z)\mathbf{p}(z)$, niin $\mathrm{syt}\big(\mathbf{a}(z),\mathbf{p}(z)\big) = \bar{1} \neq \bar{0}$ ja voidaan kirjoittaa

$$\bar{1} = \mathbf{e}_1(z)\mathbf{a}(z) + \mathbf{e}_2(z)\mathbf{p}(z).$$

Näin ollen $\mathbf{a}(z)$:lla on tällöin käänteisalkio $\mathbf{e}_1(z)$ modulo $\mathbf{p}(z)$ eli $\overline{\mathbf{a}(z)}$:lla on $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$:ssä käänteisalkio $\overline{\mathbf{e}_1(z)}$. (Samalla tuli esitettyä menetelmä, jolla käänteisalkion voi löytää.) Siispä $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$ on äärellinen kunta, jossa on $p^{\deg(\mathbf{p}(z))}$ alkiota. Huomaa, että koska $\mathbf{p}(z)$ on $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$:n nolla-alkio, z on itse asiassa $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$:n polynomirenkaassa $\mathbf{p}(z)$:n juuri. Usein merkitäänkin z:a $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$:n alkiona esimerkiksi α:lla ja kirjoitetaan $\mathbf{p}(\alpha) = \bar{0}$. Muut alkiot voidaankin sitten kirjoittaa muotoon

$$c_1 + c_2\alpha + c_3\alpha^2 + \cdots + c_n\alpha^{n-1},$$

missä $n = \deg(\mathbf{p}(z))$ ja kertoimet $c_1, c_2, \ldots, c_n$ ovat $\mathbb{Z}_p$:n alkioita, eli oleellisesti n-vektoreiksi, joiden komponentit ovat $\mathbb{Z}_p$:ssä.

Voidaan näyttää (sivuutetaan tässä, ks. kurssi 73115 Algebra 1 tai esimerkiksi ADÁMEK), että kaikki äärelliset kunnat saadaan tällä tavoin. (Myös alkukunta $\mathbb{Z}_p$ itse, kun valitaan $\mathbf{p}(z)$:ksi vaikkapa jaoton polynomi $z + \bar{1}$.) Äärellisen kunnan alkioiden lukumäärä on siis aina alkuluvun potenssi. Äärellisiä kuntia voidaan konstruoida monella tavalla, samaa astetta olevia jaottomia $\mathbb{Z}_p[z]$:n polynomejakin löytyy yleensä useampia, mutta aina äärellinen kunta, jossa on p^n alkiota, on rakenteeltaan samanlainen eli isomorfinen jonkin kunnan $\mathbb{Z}_p[z]/\langle\mathbf{p}(z)\rangle$:n kanssa, missä $\deg(\mathbf{p}(z)) = n$. Näin ollen äärellisiä kuntia, joissa on p^n alkiota, on vain yksi, ja sitä merkitään $\mathrm{GF}(p^n)$:llä. Jokaista mahdollista alkuluvun potenssia p^n kohti on edelleen olemassa $\mathrm{GF}(p^n)$, ts. jokaisesta polynomirenkaasta $\mathbb{Z}_p[z]$ löytyy kaikkia astelukuja $n \geq 1$ olevia jaottomia polynomeja.

Esimerkiksi $\mathrm{GF}(2^6)$:n konstruoimiseksi voidaan valita $\mathbb{Z}_2[z]$;n astetta 6 oleva jaoton polynomi $\mathbf{p}(z) = z^6 + z + 1$. Tarkistetaan Maplella, että $\mathbf{p}(z)$ todella on jaoton:

```
> Irreduc(z^6+z+1) mod 2;
```

true

Maplen GF-paketilla voidaan laskea äärellisissä kunnissa, tosin vähän kankeasti. Kokeillaan sitä GF(2^6):ssä:

```
> readlib(GF):
> G64:=GF(2,6,alpha^6+alpha+1):
> a:=G64[ConvertIn](alpha);
```

$a := 10000$

```
> b:=G64[`^`](a,2);
```

$b := 100000000$

```
> G64[ConvertOut](b);
```

α^2

```
> c:=G64[inverse](a);
```

$c := 10000000000000000001$

```
> G64[ConvertOut](c);
```

$\alpha^5 + 1$

```
> G64[`+`](a,G64[`^`](c,39));
```

10000000000010001

```
> G64[ConvertOut](");
```

$\alpha^4 + \alpha + 1$

Maplella on oma esitysmuotonsa kunnan alkioille, johon/josta konvertoidaan käskyillä ConvertIn/ConvertOut. Ellei tiedä yhtään sopivaa jaotonta $\mathbb{Z}_p$[z]:n polynomia, etsii Maple kyllä jonkin sellaisen:

```
> readlib(GF):
> G81:=GF(3,4):
> G81[extension];
```

10001000200020002

```
> G81[ConvertOut](");
```

$?^4 + ?^3 + 2\,?^2 + 2\,? + 2$

Valinnan saa selville extension-käskyllä, tässä se oli $\mathbb{Z}_3$[z]:n jaoton polynomi $z^4 + z^3 + \overline{2}z^2 + \overline{2}z + \overline{2}$.

Koska äärellisessä kunnassa on vain äärellinen määrä alkioita, on sen alkion β erilaisia potensseja β^j vain tietty äärellinen määrä. Ko. lukumäärä on β:n *kertaluku*, merkitään ord(β). Voidaan osoittaa, että jokaisessa äärellisessä kunnassa GF(p^n) on alkioita, joiden kertaluku on suurin mahdollinen eli $p^n - 1$, ks. kurssi 73115 Algebra 1. Tällaisia alkioita sanotaan GF(p^n):n *primitiivisiksi alkioiksi.* Jos γ on GF(p^n):n primitiivinen alkio, voidaan jokainen muu GF(p^n):n alkio, nolla-alkiota lukuunottamatta, esittää γ:n potenssina ja kunnan alkiot ovat siis

$$\overline{0},\ \overline{1},\ \gamma,\ \gamma^2,\ \ldots,\ \gamma^{p^n-2}.$$

Tästä on se seuraus, että jokainen GF(p^n):n alkio β on polynomin $z^{p^n} - z$ juuri. Tapaus $\beta = \overline{0}$ on selvä ja jos $\beta = \gamma^i$, niin

$$\beta^{p^n} = \gamma^{ip^n} = \gamma^i(\gamma^{p^n-1})^i = \gamma^i = \beta.$$

Jokainen muu alkio kuin nolla-alkio on edelleen polynomin $z^{p^n-1} - \overline{1}$ juuri. Jokainen GF(p^n):n alkio β on edellä olevan nojalla jonkin $\mathbb{Z}_p[z]$:n polynomin juuri. Alinta astetta olevaa tällaista $\mathbb{Z}_p[z]$:n polynomia kutsutaan β:n *minimipolynomiksi*. Minimipolynomi on $\mathbb{Z}_p[z]$:n jaoton polynomi, muutoin β olisi sen jonkin tekijän juuri. Usein valitaan minimipolynomiksi pääpolynomi. Edelleen β:n minimipolynomin aste on enintään n. Jos nimittäin kirjoitetaan β:n potenssit $\overline{1},\beta,\beta^2,\ldots,\beta^n$ muotoon

$$\beta^i = c_{i1} + c_{i2}\alpha + c_{i3}\alpha^2 + \cdots + c_{in}\alpha^{n-1} \qquad (i = 0,\ldots,n),$$

niin kertoimista saadut $n + 1$ vektoria $(c_{i1},c_{i2},c_{i3},\ldots,c_{in})$ $(i = 0,\ldots,n)$ ovat lineaarisesti riippuvat ja näin löytyy sellaiset kertoimet $e_1,\ldots,e_{n+1}$, jotka eivät kaikki ole nolla-alkioita, että $e_1 + e_2\beta + e_3\beta^2 + \cdots + e_{n+1}\beta^n = \overline{0}$. Vielä todetaan, että β:n minimipolynomi $\mathbf{m}(z)$ on tekijänä jokaisessa $\mathbb{Z}_p[z]$:n polynomissa $\mathbf{n}(z)$, jonka juuri β on. Tämä paljastuu jaettaessa $\mathbf{n}(z)$ $\mathbf{m}(z)$:lla

$$\mathbf{n}(z) = \mathbf{q}(z)\mathbf{m}(z) + \mathbf{j}(z), \quad \deg(\mathbf{j}(z)) < \deg(\mathbf{m}(z)),$$

sillä tällöin β on myös $\mathbf{j}(z)$:n juuri ja näin $\mathbf{j}(z)$:n on oltava nollapolynomi. Erityisesti minimipolynomit ovat polynomin $z^{p^n} - z$ tekijöitä.

Minimipolynomeilla on aivan keskeinen rooli BCH-koodien konstruktiossa.

Maple osaa testata onko annettu alkio primitiivinen:

```
> G64[isPrimitiveElement](a);
```

true

Tässä sattui GF(2^6):n generoinnissa käytetty alkio α olemaan primitiivinen. Näin käy, jos käyttää generoinnissa jaottomana polynomina $\mathbf{p}(z)$ jonkin primitiivisen alkion minimipolynomia. Esimerkki toisesta tilanteesta:

```
> G9:=GF(3,2,alpha^2+1):
> a:=G9[ConvertIn](alpha);
```

a := 10000

```
> G9[isPrimitiveElement](a);
```

false

Maplella voi etsiä primitiivisen alkion:

```
> b:=G9[PrimitiveElement]();
```

$$b := 20001$$

```
> G9[ConvertOut](b);
```

$$2\,\alpha + 1$$

Kunnassa GF(p^n) on summien ja polynomien p:nteen potenssiin korottaminen hyvin helppoa. Binomikerroin

$$\binom{p}{i} = \frac{p(p-1)\cdots(p-i+1)}{1\cdot 2\cdots i}$$

on nimittäin jaollinen p:llä, kun i = 1,...,p – 1. Näin ollen

$$(\beta+\gamma)^p = \sum_{i=0}^{p}\binom{p}{i}\beta^i\gamma^{p-i} = \beta^p + \gamma^p,$$

mikä pätee myös polynomeille. Käyttäen kaavaa toistuvasti saadaan myös useampitermisten summien p:nnet potenssit samalla tavalla, esimerkiksi

$$(\beta + \gamma + \delta)^p = \beta^p + (\gamma + \delta)^p = \beta^p + \gamma^p + \delta^p.$$

Toisaalta alkukunnan $\mathbb{Z}_p$ alkiot $\bar{i}$ ovat polynomin $z^p - z$ juuria, joten niille $\bar{i}^p = \bar{i}$. Siispä $\mathbb{Z}_p[z]$:n polynomeille **a**(z) kaava on vieläkin yksinkertaisempi:

$$\mathbf{a}(z)^p = \mathbf{a}(z^p).$$

Toistuvat p:nteen potenssiin korotukset johtavat edelleen kaavoihin

$$(\beta + \gamma)^{p^j} = \beta^{p^j} + \gamma^{p^j} \text{ ja} \qquad \mathbf{a}(z)^{p^j} = \mathbf{a}(z^{p^j}) \qquad (j = 0,1,\ldots).$$

Jos nyt β on $\mathbb{Z}_p[z]$:n polynomin **a**(z) juuri GF(p^n):ssä, niin samoin ovat sen potenssit β^{p^j} (j = 1,2,...).

Äärellisissä kunnissa pätee $\mathbb{R}[z]$:stä tuttu tulos (todistuskin on sama):

APULAUSE. Jos β on polynomin **a**(z) juuri GF(p^n):ssä, niin z – β jakaa **a**(z):n GF(p^n):n polynomirenkaassa. **a**(z):n eri juuria on näin ollen enintään t = deg(**a**(z)) kpl ja jos eri juuria $\beta_1,\ldots,\beta_t$ on nimenomaan t kpl, niin voidaan kirjoittaa

$$\mathbf{a}(z) = \gamma(z - \beta_1)\cdots(z - \beta_t),$$

missä γ on GF(p^n):n alkio. ■

Eräs seuraus Apulauseesta on, että jos GF(p^n) on $\{\bar{0}, \bar{1}, \beta_1,\ldots,\beta_{p^n-2}\}$, niin

$$z^{p^n} - z = z(z - \overline{1})(z - \beta_1)\cdots(z - \beta_{p^n - 2}),$$

ja myös, että

$$z^p - z = z(z - \overline{1})\cdots(z - \overline{p-1}).$$

LAUSE 7. Jos GF(p^n):n alkion β minimipolynomi (pääpolynomi) on $\mathbf{m}(z)$ ja $d = \deg(\mathbf{m}(z))$, ovat $\beta,\beta^p,\beta^{p^2},\ldots,\beta^{p^{d-1}}$ sen eri juuret ja

$$\mathbf{m}(z) = (z - \beta)(z - \beta^p)(z - \beta^{p^2})\cdots(z - \beta^{p^{d-1}}).$$

Todistus. Ilmeisesti $\beta,\beta^p,\beta^{p^2},\ldots,\beta^{p^{d-1}}$ ovat $\mathbf{m}(z)$:n juuria, joten pitää näyttää, että ne ovat eri alkioita. Asetetaan vastaoletus: $\beta^{p^i} = \beta^{p^j}$, missä $i < j$, ja valitaan $j - i$ pienimmäksi mahdolliseksi. Silloin

$$\overline{0} = \beta^{p^j} - \beta^{p^i} = (\beta^{p^{j-i}} - \beta)^{p^i},$$

joten $\beta^{p^{j-i}} = \beta$. Koska $j - i$ on pienin mahdollinen, ovat potenssit $\beta,\beta^p,\beta^{p^2},\ldots,\beta^{p^{j-i-1}}$ eri alkioita. Otetaan tarkasteltavaksi astetta $j - i < d$ oleva pääpolynomi

$$\mathbf{n}_1(z) = (z - \beta)(z - \beta^p)(z - \beta^{p^2})\cdots(z - \beta^{p^{j-i-1}})$$

$$= \gamma_1 + \gamma_2 z + \cdots + \gamma_{j-i} z^{j-i-1} + z^{j-i},$$

jonka juuret nämä eri alkiot ovat, ja

$$\mathbf{n}_2(z) = \gamma_1^p + \gamma_2^p z + \cdots + \gamma_{j-i}^p z^{j-i-1} + z^{j-i}.$$

Silloin

$$\mathbf{n}_2(\beta^{p^h}) = \mathbf{n}_1(\beta^{p^{h-1}})^p = \overline{0} \qquad (h = 1,\ldots,j - i).$$

Polynomeilla $\mathbf{n}_1(z)$ ja $\mathbf{n}_2(z)$ on samat keskenään erilaiset juuret, joten yo. Apulauseen nojalla ne ovat samat polynomit. Kertoimet $\gamma_1,\gamma_2,\ldots,\gamma_{j-i}$ ovat näin ollen polynomin $z^p - z$ juuria, mutta, kuten todettiin, nämä juuret ovat $\overline{0},\overline{1},\ldots,\overline{p-1}$. Siispä $\mathbf{n}_1(z)$ on $\mathbb{Z}_p[z]$:n polynomi. Tämä on mahdotonta, sillä tällöinhän $\mathbf{m}(z)$ ei olisikaan β:n minimipolynomi. Vastaoletus on siis väärä. ■

2. BCH-koodit

Syklinen koodi, jonka koodisymbolit ovat $\mathbb{Z}_p$:n alkioita ja jonka generoijapolynomi saadaan säännöillä

(i) valitaan äärellinen kunta GF(p^m), luku n (= koodin pituus) sekä kertalukua n oleva GF(p^m):n alkio β,

(ii) muodostetaan GF(p^m) (eri) alkioiden $\beta,\beta^2,\ldots,\beta^{2t}$ minimipolynomien $\mathbf{m}_1(z),\ldots,\mathbf{m}_{2t}(z)$ (pääpolynomeja) pienin yhteinen jaettava, ts. kerrotaan minimipolynomit keskenään, mutta otetaan kukin minimipolynomi tuloon mukaan vain kerran (alkioilla $\beta,\beta^2,\ldots,\beta^{2t}$ kun voi olla samojakin minimipolynomeja; Lauseen **7** mukaan erityisesti alkioiden β^{jp}, j = 1,2,..., minimipolynomeja ei tarvitse ottaa mukaan),

(iii) valitaan saatu pienin yhteinen jaettava generoijapolynomiksi $\mathbf{g}(z)$,

on ns. *BCH-koodi** C, jonka *suunniteltu virheenkorjauskyky* on t. Jos β on GF(p^m):n primitiivinen alkio, kyseessä on *primitiivinen BCH-koodi.* Primitiivisen BCH-koodin pituus on $p^m - 1$.

Koska

$$\beta^{jn} = \overline{1} \qquad (j = 1,\ldots,2t),$$

jakavat minimipolynomit $\mathbf{m}_1(z),\ldots,\mathbf{m}_{2t}(z)$ kukin polynomin $z^n - \overline{1}$:n ja samoin tekee niiden pienin yhteinen jaettava $\mathbf{g}(z)$. Näin ollen C on todellakin syklinen koodi, jonka pituus on n (Lause **5**). Huomaa, että redundanssi on

$$\deg(\mathbf{g}(z)) = n - k \leq m(2t - \lfloor 2t/p \rfloor),$$

binääritapauksessa (p = 2) erityisesti $n - k \leq mt$.

HUOM! Yleisemmin voitaisiin ottaa generoijapolynomiksi alkioiden β^h, $\beta^{h+2},\ldots,\beta^{h+2t}$ minimipolynomien pienin yhteinen tekijä. BCH-koodi voidaan myös muodostaa käyttäen koodisymboleina mielivaltaisen äärellisen kunnan alkioita. Eräs esimerkki tällaisesta on myöhemmin käsiteltävä Reed–Solomon-koodi (ks. **III.4**).

BCH-koodi voidaan määritellä koodipolynomien juurirakenteen avulla:

LAUSE 8. Yllä oleva BCH-koodi C muodostuu tarkalleen niistä enintään astetta n – 1 olevista $\mathbb{Z}_p[z]$:n polynomeista, joiden juuria $\beta,\beta^2,\ldots,\beta^{2t}$ ovat.

Todistus. Jos $\beta,\beta^2,\ldots,\beta^{2t}$ ovat enintään astetta n – 1 olevan $\mathbb{Z}_p[z]$:n polynomin $\mathbf{c}(z)$ juuria, niin minimipolynomit $\mathbf{m}_1(z),\ldots,\mathbf{m}_{2t}(z)$ ovat $\mathbf{c}(z)$:n tekijöitä ja samoin on niiden pienin yhteinen jaettava $\mathbf{g}(z)$. Näin ollen $\mathbf{c}(z)$ on

* Tarkemmin sanoen *Bose–Ray-Chaudhuri–Hocqhuenghem-koodi,* mutta tämä nimi on liian pitkä.

koodipolynomi. Jos taas $\mathbf{c}(z)$ on koodipolynomi, niin generoijapolynomi $\mathbf{g}(z)$ on sen tekijä ja samoin ovat minimipolynomit $\mathbf{m}_1(z),\ldots,\mathbf{m}_{2t}(z)$. ■

Lauseen mukaan

$$C = \{\, \mathbf{c}(z) \mid \mathbf{c}(\beta^i) = \overline{0} \ (i = 1,\ldots,2t) \text{ ja } \deg(\mathbf{c}(z)) < n \,\},$$

ts. koodivektorit $\mathbf{c}$ ovat tarkalleen ne vektorit, jotka toteuttavat matriisiyhtälön

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}_{2t}.$$

missä $\mathbf{H}$ on matriisi

$$\mathbf{H} = \begin{pmatrix} \overline{1} & \beta & \beta^2 & \beta^3 & \cdots & \beta^{n-1} \\ \overline{1} & \beta^2 & (\beta^2)^2 & (\beta^2)^3 & \cdots & (\beta^2)^{n-1} \\ \overline{1} & \beta^3 & (\beta^3)^2 & (\beta^3)^3 & \cdots & (\beta^3)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{1} & \beta^{2t} & (\beta^{2t})^2 & (\beta^{2t})^3 & \cdots & (\beta^{2t})^{n-1} \end{pmatrix}.$$

Vaikka $\mathbf{H}$ ei olekaan **I.3**:ssa olevan tarkistusmatriisin tyyppiä (sen alkiot ovat GF(p^m):n alkioita), se kuitenkin käyttäytyy samalla tavoin ja tästä syystä $\mathbf{H}$:ta kutsutaan myös *tarkistusmatriisiksi.*

Matriisia $\mathbf{H}$ käyttäen voidaan näyttää, että BCH-koodin todellinen virheenkorjauskyky on ainakin suunniteltu. Tätä varten tarvitaan muotoa

$$\mathbf{V} = \begin{pmatrix} \overline{1} & \overline{1} & \overline{1} & \cdots & \overline{1} \\ a_1 & a_2 & a_3 & \cdots & a_\ell \\ a_1^2 & a_2^2 & a_3^2 & \cdots & a_\ell^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{\ell-1} & a_2^{\ell-1} & a_3^{\ell-1} & \cdots & a_\ell^{\ell-1} \end{pmatrix}$$

olevia matriiseja eli ns. Vandermonden matriiseja koskeva aputulos, joka pätee missä tahansa kunnassa.

APULAUSE. $\det(\mathbf{V}) = \prod_{1 \le i < j \le \ell} (a_j - a_i)$

Todistus. Ilmeisestikin kaava on oikea, kun $\ell = 2$. Jos taas ℓ on suurempi, voidaan sen arvoa toistuvasti pienentää:

$$\begin{vmatrix} \overline{1} & \overline{1} & \overline{1} & \cdots & \overline{1} \\ a_1 & a_2 & a_3 & \cdots & a_\ell \\ a_1^2 & a_2^2 & a_3^2 & \cdots & a_\ell^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{\ell-1} & a_2^{\ell-1} & a_3^{\ell-1} & \cdots & a_\ell^{\ell-1} \end{vmatrix} = \begin{vmatrix} \overline{1} & \overline{1} & \overline{1} & \cdots & \overline{1} \\ \overline{0} & a_2 - a_1 & a_3 - a_1 & \cdots & a_\ell - a_1 \\ \overline{0} & a_2^2 - a_1 a_2 & a_3^2 - a_1 a_3 & \cdots & a_\ell^2 - a_1 a_\ell \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{0} & a_2^{\ell-1} - a_1 a_2^{\ell-2} & a_3^{\ell-1} - a_1 a_3^{\ell-2} & \cdots & a_\ell^{\ell-1} - a_1 a_\ell^{\ell-2} \end{vmatrix}$$

$$= (a_2 - a_1)(a_3 - a_1)\cdots(a_\ell - a_1) \begin{vmatrix} \overline{1} & \overline{1} & \cdots & \overline{1} \\ a_2 & a_3 & \cdots & a_\ell \\ a_2^2 & a_3^2 & \cdots & a_\ell^2 \\ \vdots & \vdots & \ddots & \vdots \\ a_2^{\ell-2} & a_3^{\ell-2} & \cdots & a_\ell^{\ell-2} \end{vmatrix} = \cdots = \prod_{1 \le i < j \le \ell} (a_j - a_i). \quad \blacksquare$$

LAUSE 9. (BCH-RAJA) BCH-koodi korjaa vähintään niin monta virhettä kuin on sen suunniteltu virheenkorjauskyky.

Todistus. Näytetään, että jos BCH-koodin C suunniteltu virheenkorjauskyky on t, niin $d_{min}(C) \ge 2t + 1$. Valitaan sitä varten C:n koodivektori $\mathbf{c} \neq \mathbf{0}$, jonka paino on pienin. Jos $w(\mathbf{c}) \ge 2t + 1$, niin asia on selvä. Jos taas $w(\mathbf{c}) \le 2t$, niin valitaan $\mathbf{c}$:n nolla-alkiosta eroavat komponentit $c_{i_1},\ldots,c_{i_d}$, missä $i_1 < \cdots < i_d$ ja $d \le 2t$. Muodostetaan eo. tarkistusmatriisista $\mathbf{H}$ matriisi $\mathbf{J}$ ottamalla mukaan $\mathbf{H}$:sta vain i_1:s sarake, i_2:s sarake,... ja i_d:s sarake:

$$\mathbf{J} = \begin{pmatrix} \beta^{i_1-1} & \beta^{i_2-1} & \cdots & \beta^{i_d-1} \\ \left(\beta^2\right)^{i_1-1} & \left(\beta^2\right)^{i_2-1} & \cdots & \left(\beta^2\right)^{i_d-1} \\ \left(\beta^3\right)^{i_1-1} & \left(\beta^3\right)^{i_2-1} & \cdots & \left(\beta^3\right)^{i_d-1} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\beta^{2t}\right)^{i_1-1} & \left(\beta^{2t}\right)^{i_2-1} & \cdots & \left(\beta^{2t}\right)^{i_d-1} \end{pmatrix} = \begin{pmatrix} \beta^{i_1-1} & \beta^{i_2-1} & \cdots & \beta^{i_d-1} \\ \left(\beta^{i_1-1}\right)^2 & \left(\beta^{i_2-1}\right)^2 & \cdots & \left(\beta^{i_d-1}\right)^2 \\ \left(\beta^{i_1-1}\right)^3 & \left(\beta^{i_2-1}\right)^3 & \cdots & \left(\beta^{i_d-1}\right)^3 \\ \vdots & \vdots & \ddots & \vdots \\ \left(\beta^{i_1-1}\right)^{2t} & \left(\beta^{i_2-1}\right)^{2t} & \cdots & \left(\beta^{i_d-1}\right)^{2t} \end{pmatrix}.$$

Edellisen Apulauseen nojalla näkyy nyt suoraan, että $\mathbf{J}$:n sarakkeet ovat lineaarisesti riippumattomat. Tarkistusyhtälöstä $\mathbf{c}\mathbf{H}^T = \mathbf{0}_{2t}$ seuraa kuitenkin, että $(c_{i_1},\ldots,c_{i_d})\mathbf{J}^T = \mathbf{0}_{2t}$ ja tästä edelleen että $c_{i_1} = \cdots = c_{i_d} = \overline{0}$, mikä on mahdotonta. Näin jää jäljelle vain mahdollisuus $w(\mathbf{c}) \ge 2t + 1$. $\blacksquare$

Suure $2t + 1$ on ns. C:n *suunniteltu etäisyys*, merkitään $d_{des}(C)$.

HUOM! BCH-raja voidaan esittää yleisempänäkin kuin lauseessa on tehty, kaikkia syklisiä koodeja koskevana, ks. esimerkiksi VAN LINT.

Valitsemalla BCH-koodin määritelmässä β:ksi primitiivinen alkio saadaan

LAUSE 10. Jokaista lukua $n = p^m - 1$ ja lukua $t < n/2$ kohti on olemassa syklinen n-pituinen koodi (primitiivinen BCH-koodi), jonka koodiaakkosto on $\mathbb{Z}_p$ ja joka pystyy korjaamaan t virhettä. ■

Seuraavassa taulukossa (lähde: ANDERSON & MOHAN) on lueteltu koko joukko binäärisiä primitiivisiä BCH-koodeja parametreineen. Generoijapolynomit on annettu ns. oktaalimuodossa, ts. polynomien kerroinbitit on muunnettu kolmittain oktaaliluvuiksi ja annettu nämä luvut. Esimerkiksi 23 vastaa bittejä 010 011 eli polynomia $z^4 + z + 1$.

n	k	t	**g**(z)
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567
	36	5	1033500423
127	120	1	211
	113	2	41567
	106	3	11554743
	99	4	3447023271
	92	5	624730022327
	85	6	130704476322273
	78	7	26230002166130115
	71	9	6255010713253127753
	64	10	1206534025570773100045

Tarkistusmatriisia **H** vastaten määritellään saadun vektorin **r** *syndromivektori*

$$\mathbf{S} = (S_1,\ldots,S_{2t}) = \mathbf{r}\mathbf{H}^T = (\mathbf{r}(\beta),\mathbf{r}(\beta^2),\ldots,\mathbf{r}(\beta^{2t})).$$

Alkiot $S_1,\ldots,S_{2t}$ ovat ns. *syndromit*. Huomaa, että nämä syndromit eivät ole samoja kuin **I.3**:ssa. Koodisanan syndromit ovat kuitenkin nolla-alkioita ja näin ollen syndromit riippuvat vain virheestä, eivät kulloinkin käytetystä koodisanasta, aivan niin kuin **I.3**:ssakin.

3. Dekoodaus

Jos virheitä tapahtuu v kpl, niin koodisana **c** "vastaanotetaan" muodossa **r** = **c** + **e**, missä **e** on virhesana ja w(**e**) = v. Kirjoitetaan

$$\mathbf{e}(z) = e_{j_1} z^{j_1 - 1} + \cdots + e_{j_v} z^{j_v - 1},$$

missä $1 \le j_1 < \cdots < j_v \le n$ ja $e_{j_1}, \ldots, e_{j_v} \neq \overline{0}$. Dekoodaus on helposti suoritettavissa, kun *virhekohdat* $j_1, \ldots, j_v$ sekä *virhearvot* $e_{j_1}, \ldots, e_{j_v}$ on saatu. Tätä varten tarvitaan syndromit

$$S_i = \mathbf{e}(\beta^i) = e_{j_1} \beta^{i(j_1 - 1)} + \cdots + e_{j_v} \beta^{i(j_v - 1)} \qquad (i = 1, 2, \ldots).$$

Merkitään lyhyyden vuoksi

$$X_\ell = \beta^{j_\ell - 1} \quad \text{ja} \quad Y_\ell = e_{j_\ell} \quad (\ell = 1, \ldots, v).$$

Syndromit voidaan tällöin kirjoittaa muotoon

$$S_i = \sum_{\ell=1}^{v} Y_\ell X_\ell^i \quad (i = 1, 2, \ldots).$$

Näin muodostuu yhtälöryhmä, josta $Y_1, \ldots, Y_v$ sekä $X_1, \ldots, X_v$ pitäisi ratkaista (kun $X_1, \ldots, X_v$ on saatu, löytyvät helposti myös virhekohdat). Valitettavasti ryhmä on epälineaarinen, joten tarvitaan lisäkonsteja.

Otetaan käyttöön ns. *virhekohtapolynomi*

$$\boldsymbol{\sigma}(z) = (\overline{1} - X_1 z)(\overline{1} - X_2 z) \cdots (\overline{1} - X_v z) = \overline{1} + \sigma_1 z + \cdots + \sigma_v z^v,$$

jonka juuret ovat $X_1^{-1}, \ldots, X_v^{-1}$. Koska äärellisessä kunnassa polynomin juuret ovat helposti löydettävissä suoralla kokeilulla, riittääkin virheiden paikantamiseksi löytää kertoimet $\sigma_1, \ldots, \sigma_v$.

Ilmeisesti

$$X_\ell^v \boldsymbol{\sigma}\left(X_\ell^{-1}\right) = X_\ell^v + \sigma_1 X_\ell^{v-1} + \cdots + \sigma_v = \overline{0} \quad (\ell = 1, \ldots, v)$$

ja edelleen

$$Y_\ell X_\ell^{v+h} + \sigma_1 Y_\ell X_\ell^{v+h-1} + \cdots + \sigma_v Y_\ell X_\ell^h = \overline{0} \quad (\ell = 1, \ldots, v; h = 1, \ldots, v).$$

Lasketaan nämä yhtälöt puolittain yhteen arvoille $\ell = 1, \ldots, v$ pitäen h kiinteänä:

$$S_{v+h} + \sigma_1 S_{v+h-1} + \cdots + \sigma_v S_h = \overline{0} \qquad (h = 1, \ldots, v).$$

Näin saadaan lineaarinen yhtälöryhmä

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_v \\ S_2 & S_3 & \cdots & S_{v+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_v & S_{v+1} & \cdots & S_{2v-1} \end{pmatrix} \begin{pmatrix} \sigma_v \\ \sigma_{v-1} \\ \vdots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} -S_{v+1} \\ -S_{v+2} \\ \vdots \\ -S_{2v} \end{pmatrix},$$

josta kertoimet $\sigma_1,\ldots,\sigma_v$ voidaan ratkaista. Ryhmän matriisi $\mathbf{\Sigma}$ on muotoa

$$\mathbf{\Sigma} = \begin{pmatrix} X_1 & X_2 & \cdots & X_v \\ X_1^2 & X_2^2 & \cdots & X_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^v & X_2^v & \cdots & X_v^v \end{pmatrix} \begin{pmatrix} Y_1 & \overline{0} & \cdots & \overline{0} \\ \overline{0} & Y_2 & \cdots & \overline{0} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{0} & \overline{0} & \cdots & Y_v \end{pmatrix} \begin{pmatrix} \overline{1} & X_1 & \cdots & X_1^{v-1} \\ \overline{1} & X_2 & \cdots & X_2^{v-1} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{1} & X_v & \cdots & X_v^{v-1} \end{pmatrix},$$

kuten on laskien todettavissa. Jos nyt virheitä todella on v kpl, on matriisi ei-singulääarinen (ks. Vandermonden matriiseja koskeva Apulause s. 29) ja virhekohdat löytyvät. Jos virheitä todellisuudessa sattuukin vähemmän kuin v kpl, vastaa se tilannetta, jossa yksi tai useampi virhearvoista $Y_1,\ldots,Y_v$ on $=\overline{0}$. Tällöin ryhmän matriisi on singulääarinen.

Virhekohtien löytämiseksi menetelläänkin seuraavasti:

(1) Asetetaan $v \leftarrow t$.

(2) Jos $\mathbf{\Sigma}$ on ei-singulääarinen, ratkaistaan kertoimet $\sigma_1,\ldots,\sigma_v$ yhtälöryhmästä ja näitä käyttäen edelleen virhekohdat.

(3) Jos taas $\mathbf{\Sigma}$ on singulääarinen, asetetaan $v \leftarrow v - 1$ ja mennään kohtaan **(2)**. Jos $v = 0$, ei virheitä ole.

Binääritapauksessa kaikki virhearvot ovat =1, joten niitä ei tarvitse erikseen etsiä. Muussa tapauksessa etsitään vielä virhearvot $Y_1,\ldots,Y_v$. Ne voitaisiin ratkaista lineaarisesta yhtälöryhmästä

$$\begin{pmatrix} X_1 & X_2 & \cdots & X_v \\ X_1^2 & X_2^2 & \cdots & X_v^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^v & X_2^v & \cdots & X_v^v \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_v \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_v \end{pmatrix}$$

(jonka kerroinmatriisi on ei-singulääarinen), mutta parempikin menettely löytyy. Otetaan tätä varten käyttöön syndromijonon $S_1,S_2,\ldots$ z-muunnos, ns. *syndromisarja*

$$\mathbf{S}(z) = \sum_{i=1}^{\infty} S_i z^i$$

(formaalinen potenssisarja). Koska nyt

$$(\overline{1} - X_\ell z)\sum_{i=1}^{\infty} X_\ell^i z^i = X_\ell z,$$

kuten laskien voidaan todeta, on edelleen

$$\boldsymbol{\sigma}(z)\sum_{i=1}^{\infty} X_\ell^i z^i = X_\ell z \prod_{\substack{u=1 \\ u\neq\ell}}^{v} (\overline{1} - X_u z)$$

ja

$$\boldsymbol{\sigma}(z)\mathbf{S}(z) = \boldsymbol{\sigma}(z)\sum_{i=1}^{\infty}\left(\sum_{\ell=1}^{v} Y_\ell X_\ell^i\right) z^i = \sum_{\ell=1}^{v} Y_\ell X_\ell z \prod_{\substack{u=1 \\ u\neq\ell}}^{v} (\overline{1} - X_u z).$$

Polynomia

$$\boldsymbol{\omega}(z) = \sum_{\ell=1}^{v} Y_\ell X_\ell z \prod_{\substack{u=1 \\ u\neq\ell}}^{v} (\overline{1} - X_u z)$$

kutsutaan *virhearvopolynomiksi.* Huomaa, että sen kertoimet on helposti laskettavissa kaavaa $\boldsymbol{\omega}(z) = \boldsymbol{\sigma}(z)\mathbf{S}(z)$ käyttäen. Nyt

$$\boldsymbol{\omega}(X_h^{-1}) = Y_h \prod_{\substack{\ell=1 \\ \ell\neq h}}^{v} (\overline{1} - X_\ell X_h^{-1}),$$

joten virhearvot saadaan kaavasta (ns. *Forneyn kaava*)

$$\boxed{Y_h = \frac{\boldsymbol{\omega}(X_h^{-1})}{\displaystyle\prod_{\substack{\ell=1 \\ \ell\neq h}}^{v} (\overline{1} - X_\ell X_h^{-1})} \qquad (h = 1,\ldots,v).}$$

HUOM! Arvoa $\boldsymbol{\omega}(X_h^{-1})$ <u>ei</u> voi laskea kaavaa $\boldsymbol{\omega}(z) = \boldsymbol{\sigma}(z)\mathbf{S}(z)$ käyttäen, sillä $\mathbf{S}(X_h^{-1})$ ei ole edes määritelty ja $\boldsymbol{\sigma}(X_h^{-1}) = \overline{0}$. Koska $\deg(\boldsymbol{\omega}(z)) \leq v$, seuraa eo. yhtälöryhmä kertoimille $\sigma_1,\ldots,\sigma_v$ myös yhtälöstä $\boldsymbol{\sigma}(z)\mathbf{S}(z) = \boldsymbol{\omega}(z)$.

Virhearvojen laskun nopeuttamiseksi ei ole paljoakaan tehtävissä, mutta virhekohtien etsimiseksi on kehitetty lukuisia yo. menettelyä nopeampia

tapoja, joista mainittakoon esimerkiksi Eukleideen algoritmia käyttävä menettely (ks. ADÁMEK tai VANSTONE & VAN OORSCHOTT) sekä (äärellistä) Fourier'n muunnosta ja FFT-algoritmia käyttävät menettelyt (ks. BLAHUT ja SWEENEY). Klassinen nopea menettely on ns. *Berlekamp–Massey-algoritmi,* joka kuvataan seuraavassa.

Berlekamp–Massey-algoritmi

Tarkastelemalla aikaisemmin saatua yhtälöä

$$S_{v+h} + \sigma_1 S_{v+h-1} + \cdots + \sigma_v S_h = \overline{0} \quad (h = 1,\ldots,v).$$

voidaan todeta, että syndromijono $S_1,S_2,\ldots$ saadaan "alkuarvoista" $S_1,\ldots,S_v$ lähtien käyttäen kertalukua v olevaa rekursiokaavaa

$$S_{v+h} = -\sum_{i=1}^{v} \sigma_i S_{v+h-i},$$

jonka karakteristinen yhtälö on $z^v \boldsymbol{\sigma}(\overline{1}/z) = \overline{0}$ (ks. kurssi 73265 Symbolinen analyysi 1). Näin ajateltuna virhekohtien etsiminen tarkoittaakin rekursiokaavan etsimistä syndromijonon generoimiseksi. Tällaisia rekursiokaavoja on useita. Voihan saatu sana $\mathbf{r} = \mathbf{c} + \mathbf{e}$ tulla useamman kuin v virheen kautta jostakin toisesta koodisanasta ja tällöin syndromijonolle tulisi rekursiokaava, jonka kertaluku olisi suurempi kuin v. Jos taas virheitä todellisuudessa olikin vähemmän kuin v kpl, voi kertaluku olla myös pienempi kuin v.

Berlekamp–Massey-algoritmi etsii iteroiden alinta kertalukua olevan rekursiokaavan, joka generoi syndromijonon $S_1,\ldots,S_{2t}$. Iterointilaskuri merkitään sulutettuna yläindeksiksi. Iterointi alkaa polynomista $\boldsymbol{\sigma}^{(0)}(z) = \overline{1}$ eli rekursiokaavasta $S_h = \overline{0}$. Kun iteroinnin kuluessa on menossa polynomi $\boldsymbol{\sigma}^{(j-1)}(z)$, vastaava rekursiokaava

$$S_{L_{j-1}+h} = -\sum_{i=1}^{L_{j-1}} \sigma_i^{(j-1)} S_{L_{j-1}+h-i}$$

generoi syndromijonoa syndromiin S_{j-1} asti. Jos rekursiokaava generoi jonoa vielä seuraavaankin syndromiin S_j asti, otetaan $\boldsymbol{\sigma}^{(j)}(z) = \boldsymbol{\sigma}^{(j-1)}(z)$. Muussa tapauksessa lasketaan rekursiokaavan "ennustama" väärä syndromi

$$S'_j = -\sum_{i=1}^{L_{j-1}} \sigma_i^{(j-1)} S_{j-i}$$

ja muodostetaan erotus

$$\Delta_j = S_j - S'_j = S_j + \sum_{i=1}^{L_{j-1}} \sigma_i^{(j-1)} S_{j-i} \neq \overline{0}.$$

Nyt otetaan $L_j = \max(L_{j-1}, j-u+L_{u-1})$ ja

$$\boldsymbol{\sigma}^{(j)}(z) = \boldsymbol{\sigma}^{(j-1)}(z) - \Delta_j \Gamma_u^{-1} z^{j-u} \boldsymbol{\sigma}^{(u-1)}(z),$$

missä u:s iteraatio oli edellinen sellainen iteraatio, jolloin kertalukua piti kasvattaa, ja

$$\Gamma_u = S_u - S'_u = S_u + \sum_{i=1}^{L_{u-1}} \sigma_i^{(u-1)} S_{u-i} \neq \overline{0}.$$

Sen näyttämiseksi, että saatu uusi rekursiokaava generoi syndromijonon syndromiin S_j asti, merkitään yleisesti

$$\Gamma_\ell = S_\ell + \sum_{i=1}^{L_{u-1}} \sigma_i^{(u-1)} S_{\ell-i} \quad \text{ja} \qquad \Delta_\ell = S_\ell + \sum_{i=1}^{L_{j-1}} \sigma_i^{(j-1)} S_{\ell-i}.$$

Silloin

$$\Gamma_\ell = \begin{cases} \overline{0}, & \text{kun } \ell = L_{u-1}+1,\ldots,u-1 \\ \Gamma_u \neq \overline{0}, & \text{kun } \ell = u \end{cases} \qquad \text{ja} \qquad \Delta_\ell = \begin{cases} \overline{0}, & \text{kun } \ell = L_{j-1}+1,\ldots,j-1 \\ \Delta_j \neq \overline{0}, & \text{kun } \ell = j. \end{cases}$$

Edelleen, kun $\ell = L_j + 1,\ldots,j$,

$$S_\ell + \sum_{i=1}^{L_j} \sigma_i^{(j)} S_{\ell-i} = S_\ell + \sum_{i=1}^{L_{j-1}} \sigma_i^{(j-1)} S_{\ell-i} - \Delta_j \Gamma_u^{-1} \left(S_{\ell-j+u} + \sum_{i=1}^{L_{u-1}} \sigma_i^{(u-1)} S_{\ell-j+u-i} \right)$$

$$= \Delta_\ell - \Delta_j \Gamma_u^{-1} \Gamma_{\ell-j+u} = \left\{ \begin{matrix} \overline{0}, & \text{kun } \ell = L_j+1,\ldots,j-1 \\ \Delta_j - \Delta_j \Gamma_u^{-1} \Gamma_u, & \text{kun } \ell = j \end{matrix} \right\} = \overline{0}.$$

Siispä

$$S_\ell = -\sum_{i=1}^{L_j} \sigma_i^{(j)} S_{\ell-i} \quad (\ell = L_j+1,\ldots,j),$$

kuten pitääkin. (Tarvittaessa sovitaan, että $S_0 = \overline{1}$.)

Menettelyn tuloksena saadaan lopulta virhepolynomi $\boldsymbol{\sigma}(z) = \boldsymbol{\sigma}^{(2t)}(z)$ ja vastaava rekursiokaava. Voidaan osoittaa, että saatu rekursiokaava on alinta mahdollista kertalukua, mutta tämä ei ole aivan helppoa, ks. esimerkiksi

BLAHUT tai ANDERSON & MOHAN*· Todistuksen yhteydessä esiintyy seuraava aputulos, jolla on muutakin mielenkiintoa.

APULAUSE. Jos rekursiokaava muuttuu j:nnellä iteraatiokierroksella, niin $L_j = \max(L_{j-1}, j - L_{j-1})$.

Todistus. Kun kertaluku muuttuu ensimmäisen kerran, asia on selvä, sillä silloin $L_{j-1} = 0$ ja $L_j = j$ (kaava on $S_j = S_j \cdot S_0$ ja kertaluku kasvaa). Oletetaan sitten, että tulos on pitänyt paikkansa aikaisemmilla iteraatiokierroksilla, joilla rekursiokaavaa on pitänyt muuttaa, ja tarkastellaan seuraavaa eli j:ttä iteraatiokierrosta, jolloin taas pitää muuttaa rekursiokaavaa. Silloin $L_j = \max(L_{j-1}, j - u + L_{u-1})$. Toisaalta, koska u:nnella iteraatiokierroksella rekursiokaavan kertaluku kasvoi ja tulos piti paikkansa, niin $L_u = u - L_{u-1}$. Iteraatiokierroksilla $u,\ldots,j-1$ kertaluku ei kasva, joten myös $L_{j-1} = u - L_{u-1}$. Siispä $L_j = \max(L_{j-1}, j - L_{j-1})$. ■

Seuraavassa on Berlekamp–Massey-algoritmin pseudokoodi, jossa on käytetty yo. Apulausetta:

```
BEGIN INPUT S = (S_1,...,S_2t)
j := 0; σ(z) := 1; L := 0; β(z) := 1;
FOR j = 1 TO 2t DO
      BEGIN
      Δ := S_j + Σ_{i=1}^{L} σ_i S_{j-i};
      IF Δ ≠ 0 THEN
            BEGIN
            τ(z) := σ(z) – Δz β(z);
            IF 2L < j THEN
                  BEGIN
                  β(z) := Δ^{-1} σ(z);
                  L := j – L;
                  END;
            ELSE
                  β(z) := z β(z);
            σ(z) := τ(z);
            END;
      ELSE
            β(z) := z β(z);
      END;
IF deg(σ(z)) ≠ L THEN
      OUTPUT "Enemmän kuin t virhettä, ei voida korjata."
ELSE OUTPUT σ(z)
END.
```

* Klassinen artikkeliviite on MASSEY, J.L.: Shift Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory* **15** (1969), 122–127.

4. Reed–Solomon-koodit

Reed–Solomon-koodin koodisymbolit ovat äärellisen kunnan $\mathbb{F} = GF(p^m)$ alkioita ja generoijapolynomi on

$$\mathbf{g}(z) = (z - \beta)(z - \beta^2)\cdots(z - \beta^{2t}),$$

missä β on $\mathbb{F}$:n primitiivinen alkio ja $2t \leq p^m - 1$. Koodin pituus on $n = p^m - 1$ ja redundanssi on

$$n - k = \deg(\mathbf{g}(z)) = 2t.$$

t on *suunniteltu virheenkorjauskyky*. Koska $\beta,\beta^2,\ldots,\beta^{2t}$ ovat polynomin $z^n - \overline{1} = z^{p^m - 1} - \overline{1}$ juuria, ovat polynomit $z - \beta, z - \beta^2, \ldots, z - \beta^{2t}$ s. 26 olevan Apulauseen nojalla $z^n - \overline{1}$:n tekijöitä ja samoin on $\mathbf{g}(z)$. Näin ollen (Lause **5**) Reed–Solomon-koodi todella on syklinen koodi.

HUOM! Reed–Solomon-koodi ei ole **III.2**:ssa määritelty BCH-koodi. Toisaalta, kuten Huomautuksessa s. 28 mainittiin, BCH-koodi voitaisiin määritellä yleisemminkin, jolloin taas Reed–Solomon-koodi olisi sen erikoistapaus.

LAUSE 11. Reed–Solomon-koodin suunniteltu virheenkorjauskyky on sen tarkka virheenkorjauskyky.

Todistus. Katsotaan yllä määritellyn Reed–Solomon-koodin minimietäisyyttä d_{min}. Koska $\mathbf{g}(z)$ on koodipolynomi, on $d_{min} \leq 2t + 1$ eli (Lause **1**) virheenkorjauskyky on enintään t. Toisaalta Lauseen **10** todistus käy sellaisenaan myös Reed–Solomon-koodille, joten virheenkorjauskyky on vähintään t. Siispä se on tarkalleen t. ■

Edellä esitetty BCH-koodien dekoodausmenettely sopii aivan sellaisenaan Reed–Solomon-koodeille. Erityisesti Berlekamp–Massey-algoritmi käy sellaisenaan sekä myös Forneyn kaava

$$Y_h = \frac{\boldsymbol{\omega}(X_h^{-1})}{\prod_{\substack{\ell=1 \\ \ell \neq h}}^{v} (\overline{1} - X_\ell X_h^{-1})} \qquad (h = 1,\ldots,v).$$

Tulkitsemalla Reed–Solomon-koodin koodisymbolit $\mathbb{Z}_p$:n alkioiden muodostamiksi m-vektoreiksi saadaan mn-pituinen lineaarinen koodi, jonka koodisymbolit ovat $\mathbb{Z}_p$:n alkioita:

$$\underbrace{\overbrace{a_1 a_2 \cdots a_m}^{= \mathbf{a}_1}}_{= \alpha_1} \underbrace{\overbrace{a_{m+1} \cdots a_{2m}}^{= \mathbf{a}_2}}_{= \alpha_2} \cdots \underbrace{\overbrace{a_{(m-1)n+1} \cdots a_{mn}}^{= \mathbf{a}_n}}_{= \alpha_n}.$$

Vaikka binääristä Reed–Solomon-koodia ei olekaan, saadaan tällä tavoin binäärinen koodi, kun valitaan $\mathbb{F} = GF(2^m)$ ja $m \geq 2$.

Kun ajatellaan virheiden olevan $\mathbb{Z}_p$:n alkioissa vektoreissa $\mathbf{a}_1,\ldots,\mathbf{a}_n$, ei tästä konstruktiosta ole iloa yksittäisten virheiden korjaamisen kannalta. Toisin on asianlaita, jos virheet esiintyvätkin purskeina, ts. virheitä on paljon, mutta ne ovat vain enintään t:ssä vektoreista $\mathbf{a}_1,\ldots,\mathbf{a}_n$. Tällaiset purskevirheet voidaan korjata tulkitsemalla vektorit $\mathbf{a}_1,\ldots,\mathbf{a}_n$ takaisin Reed–Solomon-koodin koodisymboleiksi ja dekoodaamalla. Reed–Solomon-koodeja käytetään paljon myös mutkikkaampien purskevirheitä korjaavien koodien konstruktiossa (ks. seuraava luku).

5. Pyyhkiymien korjaus

Tilanteissa, joissa BCH-koodia tai Reed–Solomon-koodia käytetään, esiintyy usein “tavallisten” virheiden lisäksi pyyhkiymiä. Jos koodin minimietäisyys on d_{min}, niin u virhettä ja w pyyhkiymää voidaan korjata yhtaikaisesti tarkalleen silloin, kun

$$2u + w + 1 \leq d_{min}$$

(ks. s. 3). Tarkastellaankin nyt BCH- tai Reed–Solomon-koodia, jonka suunniteltu etäisyys on $d_{des} = 2t + 1$, ja oletetaan, että $2u + w + 1 \leq d_{des}$. Syndromien laskua varten korvataan pois pyyhkiytyneet symbolit $\overline{0}$:lla (mitkä tahansa muutkin symbolit kävisivät). Virhekohdat ovat $j_1,\ldots,j_u$ ja *pyyhkiymäkohdat* $h_1,\ldots,h_w$. Virhearvot ovat $e_{j_1},\ldots,e_{j_u},e_{h_1},\ldots,e_{h_w}$ (pyyhkiymille saadaan virhearvot, koska pois pyyhkiytyneet symbolit korvattiin $\overline{0}$:lla). Merkitään lyhyyden vuoksi

$$X_\ell = \beta^{j_\ell - 1} \quad \text{ja} \quad Y_\ell = e_{j_\ell} \quad (\ell = 1,\ldots,u)$$

sekä

$$U_\ell = \beta^{h_\ell - 1} \quad \text{ja} \quad Z_\ell = e_{h_\ell} \quad (\ell = 1,\ldots,w).$$

Virhesyndromit ovat silloin

$$T_i = \sum_{\ell=1}^{u} Y_\ell X_\ell^i \quad (i = 1,2,\ldots)$$

ja *pyyhkiymäsyndromit*

$$R_i = \sum_{\ell=1}^{w} Z_\ell U_\ell^i \quad (i = 1,2,\ldots).$$

Varsinaiset syndromit ovat

$$S_i = T_i + R_i \qquad (i = 1,2,\ldots).$$

Merkitään nyt $v = u + w$ ja määritellään edelleen virhekohtapolynomi

$$\boldsymbol{\tau}(z) = (\overline{1} - X_1 z)(\overline{1} - X_2 z)\cdots(\overline{1} - X_u z) = \overline{1} + \tau_1 z + \cdots + \tau_u z^u,$$

jonka juuret ovat $X_1^{-1},\ldots,X_u^{-1}$, sekä *pyyhkiymäpolynomi*

$$\boldsymbol{\rho}(z) = (\overline{1} - U_1 z)(\overline{1} - U_2 z)\cdots(\overline{1} - U_w z) = \overline{1} + \rho_1 z + \cdots + \rho_w z^w,$$

jonka juuret ovat $U_1^{-1},\ldots,U_w^{-1}$. Huomaa, että koska pyyhkiymien paikat tiedetään, on $\boldsymbol{\rho}(z)$ täysin tunnettu. Vielä merkitään

$$\boldsymbol{\sigma}(z) = \boldsymbol{\tau}(z)\boldsymbol{\rho}(z) = \overline{1} + \sigma_1 z + \cdots + \sigma_v z^v.$$

Lienee mukavinta johtaa $\boldsymbol{\tau}(z)$:n kertoimien etsimiseen tarvittava yhtälöryhmä käyttäen erilaisia syndromisarjoja (ks. Huomautus s. 34). Merkitään tätä varten

$$\mathbf{T}(z) = \sum_{i=1}^{\infty} T_i z^i \qquad \text{ja} \qquad \mathbf{R}(z) = \sum_{i=1}^{\infty} R_i z^i$$

(*virhesyndromisarja* ja *pyyhkiymäsyndromisarja*). Tällöin

$$\mathbf{S}(z) = \mathbf{T}(z) + \mathbf{R}(z).$$

Nyt

$$\boldsymbol{\rho}(z)\mathbf{S}(z) = \boldsymbol{\rho}(z)\mathbf{T}(z) + \boldsymbol{\epsilon}(z),$$

missä $\boldsymbol{\epsilon}(z) = \boldsymbol{\rho}(z)\mathbf{R}(z)$ on enintään astetta w oleva polynomi (vrt. **III.3**:n vastaava tulos $\boldsymbol{\sigma}(z)\mathbf{S}z) = \boldsymbol{\omega}(z)$). Edelleen

$$\boldsymbol{\sigma}(z)\mathbf{S}(z) = \boldsymbol{\sigma}(z)\mathbf{T}(z) + \boldsymbol{\tau}(z)\boldsymbol{\epsilon}(z) = \boldsymbol{\omega}(z),$$

missä $\deg(\boldsymbol{\tau}(z)\boldsymbol{\epsilon}(z)) \le v$ ja $\deg(\boldsymbol{\omega}(z)) \le v$. Siispä, kun merkitään $\sigma_0 = \overline{1}$,

$$\sum_{j+\ell=i} \sigma_j T_\ell = \overline{0} \quad (i = v+1, v+2,\ldots),$$

josta saadaan yhtälöryhmä

$$\begin{pmatrix} T_1 & T_2 & \cdots & T_{v+1} \\ T_2 & T_3 & \cdots & T_{v+2} \\ \vdots & \vdots & \ddots & \vdots \\ T_u & T_{u+1} & \cdots & T_{v+u} \end{pmatrix} \begin{pmatrix} \sigma_v \\ \vdots \\ \sigma_1 \\ \overline{1} \end{pmatrix} = \mathbf{0}_u$$

(yhtälöitä on u kpl, sillä varsinaisesti ei nyt olla ratkaisemassa kertoimia $\sigma_1,\ldots,\sigma_v$, vaan kertoimia $\tau_1,\ldots,\tau_u$). Ryhmän matriisi $\mathbf{\Psi}$ on muotoa

$$\mathbf{\Psi} = \begin{pmatrix} X_1 & X_2 & \cdots & X_u \\ X_1^2 & X_2^2 & \cdots & X_u^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^u & X_2^u & \cdots & X_u^u \end{pmatrix} \begin{pmatrix} Y_1 & \overline{0} & \cdots & \overline{0} \\ \overline{0} & Y_2 & \cdots & \overline{0} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{0} & \overline{0} & \cdots & Y_u \end{pmatrix} \begin{pmatrix} \overline{1} & X_1 & \cdots & X_1^v \\ \overline{1} & X_2 & \cdots & X_2^v \\ \vdots & \vdots & \ddots & \vdots \\ \overline{1} & X_u & \cdots & X_u^v \end{pmatrix}.$$

Huomaa, että tulon ensimmäinen matriisi on ei-singulaarinen (ks. Vandermonden matriiseja koskeva Apulause s. 29) ja että tulon toinen matriisi on sekin ei-singulaarinen, jos $Y_1,\ldots,Y_u \neq \overline{0}$.

Toisaalta kertoimet $\sigma_1,\ldots,\sigma_v$ saadaan kertoimista $\tau_1,\ldots,\tau_u$ yhtälöllä

$$\begin{pmatrix} \sigma_v \\ \vdots \\ \sigma_1 \\ \overline{1} \end{pmatrix} = \mathbf{\Theta} \begin{pmatrix} \tau_u \\ \vdots \\ \tau_1 \\ \overline{1} \end{pmatrix},$$

missä

$$\mathbf{\Theta} = \begin{pmatrix} \rho_w & \overline{0} & \cdots & \overline{0} & \overline{0} & \overline{0} \\ \rho_{w-1} & \rho_w & \cdots & \overline{0} & \overline{0} & \overline{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \rho_1 & \rho_2 & \cdots & \rho_{u-1} & \rho_u & \rho_{u+1} \\ \overline{1} & \rho_1 & \cdots & \rho_{u-2} & \rho_{u-1} & \rho_u \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \overline{0} & \overline{0} & \cdots & \overline{1} & \rho_1 & \rho_2 \\ \overline{0} & \overline{0} & \cdots & \overline{0} & \overline{1} & \rho_1 \\ \overline{0} & \overline{0} & \cdots & \overline{0} & \overline{0} & \overline{1} \end{pmatrix}$$

(tarpeen mukaan merkitään $\rho_{w+1} = \rho_{w+2} = \cdots = \overline{0}$). Näin ollen käyttäen matriisia $\mathbf{\Psi\Theta}$ kertoimille $\tau_1,\ldots,\tau_u$ saadaan lineaarinen yhtälöryhmä

$$\mathbf{\Psi\Theta} \begin{pmatrix} \tau_u \\ \vdots \\ \tau_1 \\ \overline{1} \end{pmatrix} = \mathbf{0}_u,$$

josta ne yksikäsitteisesti ratkeavat (olettaen siis, että $Y_1,\ldots,Y_u \neq \overline{0}$). Tämä voidaan todeta laskemalla

$$\begin{pmatrix} \overline{1} & X_1 & \cdots & X_1^v \\ \overline{1} & X_2 & \cdots & X_2^v \\ \vdots & \vdots & \ddots & \vdots \\ \overline{1} & X_u & \cdots & X_u^v \end{pmatrix} \boldsymbol{\Theta} = \begin{pmatrix} X_1^w \rho(X_1^{-1}) & X_1^{w+1} \rho(X_1^{-1}) & \cdots & X_1^{w+u} \rho(X_1^{-1}) \\ X_2^w \rho(X_2^{-1}) & X_2^{w+1} \rho(X_2^{-1}) & \cdots & X_2^{w+u} \rho(X_2^{-1}) \\ \vdots & \vdots & \ddots & \vdots \\ X_u^w \rho(X_u^{-1}) & X_u^{w+1} \rho(X_u^{-1}) & \cdots & X_u^{w+u} \rho(X_u^{-1}) \end{pmatrix}$$

ja soveltamalla Apulausetta sivulla 29, sillä $\boldsymbol{\rho}(X_\ell^{-1}) \neq \overline{0}$ ($\ell = 1,\ldots,u$).

Kerroinmatriisin saamiseksi otetaan nyt käyttöön jo aikaisemmin esiintynyt ns. *kvasisyndromisarja*

$$\mathbf{Q}(z) = \boldsymbol{\rho}(z)\mathbf{S}(z) = \boldsymbol{\rho}(z)\mathbf{T}(z) + \boldsymbol{\epsilon}(z) = \sum_{i=1}^{\infty} Q_i z^i.$$

Tässä esiintyvät ns. *kvasisyndromit* $Q_1, Q_2, \ldots, Q_{2t}$ ovat täysin tunnettuja ja, koska $\deg(\boldsymbol{\epsilon}(z)) \leq w$,

$$Q_i = \sum_{j+\ell=i} \rho_j T_\ell \quad (i = w+1, w+2\ldots).$$

(Merkitään $\rho_0 = \overline{1}$.) Mutta tämä tarkoittaa, että eo. yhtälöryhmän kerroinmatriisi

$$\boldsymbol{\Psi\Theta} = \begin{pmatrix} Q_{w+1} & Q_{w+2} & \cdots & Q_{w+u+1} \\ Q_{w+2} & Q_{w+3} & \cdots & Q_{w+u+2} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{w+u} & Q_{w+u+1} & \cdots & Q_{w+2u} \end{pmatrix}$$

saadaankin kvasisyndromeista $Q_{w+1}, Q_{w+2}, \ldots, Q_{w+2u}$ eli se on tunnettu. (Muista, että $w + 2u \leq 2t$.) Virhekohtien etsintä on näin aivan samanlaista kuin se oli **III.3**:ssa, syndromien $S_1, S_2, \ldots, S_{2v}$ sijasta käytetään vain kvasisyndromeja $Q_{w+1}, \ldots, Q_{w+2u}$. Myöskin Berlekamp–Massey-algoritmi voidaan ottaa sellaisenaan käyttöön.

Virhearvot tarvitaan sekä “tavallisille” virheille että pyyhkiymille. Pyyhkiymän virhearvo saattaa olla $\overline{0}$:kin, sillä korvattaessa pois pyyhkiytyneet symbolit $\overline{0}$:illa jotkin pyyhkiymät jo ehkä korjautuivat (ei vain tiedetä kävikö näin ja mitkä pyyhkiymät korjautuivat). Virhearvojen lasku on aivan samanlaista kuin aikaisemmin **III.3**.ssa.

IV LUKU
PURSKEVIRHEIDEN KORJAUS

1. Yleistä

Jos koodisanassa $\mathbf{c} = c_1c_2\cdots c_n$ alusta lukien ensimmäinen virhe tapahtuu symbolissa c_i ja viimeinen virhe symbolissa c_j, niin virheet muodostavat *purskevirheen,* jonka pituus on $j - i + 1$. Jos purskevirheen pituus on pieni, niin se voi olla helpommin korjattavissa kuin esiintyvät yksittäiset virheet, vaikka purske sisältäisi virheitä paljonkin.

ESIMERKKI. Toistokoodi (ks. Esimerkki s. 1) pystyy ilmeisesti korjaamaan purskevirheen, mikäli purske ulottuu vain enintään $\lfloor (r-1)/2 \rfloor$ peräkkäiseen toistettuun viestisanaan. Ts. enintään $(\lfloor (r-1)/2 \rfloor - 1)k + 1$-pituiset purskevirheet voidaan ainakin korjata. Mutta tarkemmin mietittäessä voi toistokoodin purskevirheenkorjauskyvyn todeta vielä paremmaksi: enintään $\lfloor (r-1)/2 \rfloor k$-pituiset purskevirheet voidaan korjata, vaikkakaan ei enää kaikkia $\lfloor (r-1)/2 \rfloor k + 1$-pituisia purskevirheitä. Esimerkiksi binäärinen toistokoodi $\{000000, 010101, 101010, 111111\}$ $(r = 3,\ k = 2)$ pystyy korjaamaan 2-pituiset purskevirheet.

Lineaarisille koodeille saadaan

LAUSE 12. (REIGER-RAJA)

(i) Jos lineaarinen (n,k)-koodi paljastaa enintään d-pituiset purskevirheet, niin $n - k \geq d$.

(ii) Jos lineaarinen (n,k)-koodi korjaa enintään d-pituiset purskevirheet, niin $n - k \geq 2d$.

Todistus. **(i)** Oletetaan, että lineaarinen (n,k)-koodi C, jonka koodisymbolit ovat kunnan $\mathbb{F}$ alkioita, paljastaa enintään d-pituiset purskevirheet. $\mathbb{F}^n$:n sellaisia vektoreita, joiden $n - d$ viimeistä komponenttia ovat $= \overline{0}$, on q^d kpl, missä q on kunnan $\mathbb{F}$ alkioluku. Näiden täytyy kuulua eri sivuluokkiin (ks. **I.5**). Jos nimittäin **a** ja **b** ovat tällaisia vektoreita eivätkä ole sa-

mat, niin $\mathbf{a} - \mathbf{b}$ on enintään d-pituinen purskevirhevektori eikä näin ollen ole koodisana. Sivuluokkia on q^{n-k} kpl, joten $n - k \geq d$.

(ii) Oletetaan, että lineaarinen (n,k)-koodi C korjaa enintään d-pituiset purskevirheet. Jokainen enintään 2d-pituinen (ja vähintään 2-pituinen) purskevirhevektori $\mathbf{e}$ voidaan kirjoittaa kahden enintään d-pituisen purskevirhevektorin erotuksena: $\mathbf{e} = \mathbf{e}_1 - \mathbf{e}_2$. Jos tällöin $\mathbf{e}$ olisi koodivektori, niin C ei pystyisi korjaamaan kaikkia enintään d-pituisia purskevirheitä, sillä $\mathbf{0} + \mathbf{e}_1 = \mathbf{e} + \mathbf{e}_2$. 1-pituiset purskevirheet C tietysti korjaakin ilman muuta. Näin ollen kaikki enintään 2d-pituiset purskevirheet voidaan ainakin paljastaa, jolloin **(i)**-kohdan nojalla $n - k \geq 2d$. ■

Lineaarisia (n,k)-koodeja, joiden purskevirheenkorjauskyky yltää Reiger-rajaan $(n - k)/2$, kutsutaan *Reiger-optimaalisiksi.* Jos lineaarinen (n,k)-koodi korjaa kaikki d-pituiset purskevirheet, mutta ei enää kaikkia d + 1-pituisia, niin sen *purskeenkorjaussuhde* on $2d/(n - k)$. Reiger-optimaalisten koodien purskeenkorjaussuhde on siis 1.

ESIMERKKI. (Jatkoa) Toistokoodi on Reiger-optimaalinen, jos r on pariton. Parilliselle arvolle $r = 2s$, purskeenkorjaussuhde on

$$\frac{2\lfloor (2s-1)/2 \rfloor k}{2sk - k} = \frac{2s-2}{2s-1} = \frac{r-2}{r-1} = 1 - \frac{1}{r-1}.$$

ESIMERKKI. III.4:ssä mainittu Reed–Solomon-koodista johdettu koodi on lineaarinen koodi, jonka koodisymbolit ovat alkukunnan $\mathbb{Z}_p$ alkiot. Se korjaa purskevirheet, jotka eivät koske useampaa kuin t:tä peräkkäistä vastaavan Reed–Solomon-koodin symbolia. Näin ollen enintään $(t - 1)m + 1$-pituiset purskevirheet voidaan korjata. Purskeenkorjaussuhde on siis (ainakin)

$$\frac{2((t-1)m+1)}{2tm} = 1 - \frac{m-1}{tm}.$$

Lineaarisille (n,k)-koodeille saadaan Reiger-rajan lisäksi muodostetuksi toinenkin raja. Jos koodi korjaa enintään d-pituiset purskevirheet, niin yhdessä sivuluokassa ei saa olla useampia tällaisia purskevirhevektoreita (muuten kahden tällaisen vektorin erotus on koodivektori). 1-pituisia purskevirhevektoreita on ilmeisesti $n(q - 1)$ kpl ja t-pituisia $(t \geq 2)$ purskevirhevektoreita on toisaalta $(n - t + 1)(q - 1)^2 q^{t-2}$ kpl. Lasketaan Maplella enintään d-pituisten purskevirheiden lukumäärä:

```
> simplify(sum((n-t+1)*q^(t-2),t=2..d)*(q-1)^2+n*(q-1));
```

$$q^d n - q^{(d-1)} n + q^d - q^d d + q^{(d-1)} d - 1$$

Näin ollen kyseinen lukumäärä on $((n-d)(q-1)+q)q^{d-1}-1$. Sivuluokkia on q^{n-k} kpl (joista yksi muodostuu koodisanoista), joten saadaan

LAUSE 13. Jos lineaarinen (n,k)-koodi korjaa enintään d-pituiset purskevirheet, niin

$$n-k \geq \log_q((n-d)(q-1)+q) + d - 1. \quad \blacksquare$$

Sykliselle koodille saadaan myös purskevirheenpaljastuskyvyn alaraja:

LAUSE 14. Syklinen (n,k)-koodi paljastaa enintään n – k-pituiset purskevirheet.

Todistus. Tarkastellaan koodin generoijapolynomia **g**(z). Degeneroitunut tapaus **g**(z) = $\overline{1}$ on selvä (n – k = 0). Siirrytään tapaukseen **g**(z) ≠ $\overline{1}$. Jos purskevirheen pituus on d ≤ n – k, niin vastaava purskevirhepolynomi on muotoa z^i**a**(z), missä deg(**a**(z)) ≤ d – 1 ≤ n – k – 1. Koska toisaalta deg(**g**(z)) = n – k, niin **g**(z) ei voi olla **a**(z):n tekijä eikä näin ollen myöskään z^i**a**(z):n tekijä. Purskevirhepolynomi ei siis ole koodipolynomi, mikä paljastaa sen. ■

Syklisen koodin purskevirheitä korjaava dekoodaus voidaan toteuttaa Meggitt-dekoodauksena (ks. **II.6**) käyttäen syndromitaulua, jossa ovat kaikkien niiden (korjattavissa olevien) purskevirheiden syndromit, missä yksi purskeen virheistä tapahtuu viimeisessä koodisanan symbolissa.

Seuraavassa taulukossa (lähde: ANDERSON & MOHAN) on annettu eräitä binäärisiä Reiger-optimaalisia syklisiä koodeja. Generoijapolynomit **g**(z) on annettu oktaalimuodossa, ks. s. 31. Koodit on etsitty tietokoneella.

d	n	k	**g**(z)
2	7	3	35
3	15	9	171
4	15	7	721
	19	11	1151
5	15	5	2467
	27	17	2671
6	21	9	14515
	34	22	15173
7	21	7	47343
	38	24	114361
8	21	5	214537
	50	34	224531
9	21	3	1647235
	56	38	1505773
10	59	39	40003351

2. Tulokoodit

Tulokoodeja muodostettaessa käytetty “tulo” on ns. Kronecker-tulo. Matriisien **A** ja **B** *Kronecker-tulo* $\mathbf{A} \otimes \mathbf{B}$ määritellään seuraavasti: Jos **A** on $n_1 \times m_1$-matriisi

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1m_1} \\ \vdots & \ddots & \vdots \\ a_{n_1 1} & \cdots & a_{n_1 m_1} \end{pmatrix}$$

ja **B** on $n_2 \times m_2$-matriisi, niin (lohkomuodossa) $\mathbf{A} \otimes \mathbf{B}$ on $n_1 n_2 \times m_1 m_2$-matriisi

$$\mathbf{A} \otimes \mathbf{B} = \left(\begin{array}{c|c|c} a_{11}\mathbf{B} & \cdots & a_{1m_1}\mathbf{B} \\ \hline \vdots & \ddots & \vdots \\ \hline a_{n_1 1}\mathbf{B} & \cdots & a_{n_1 m_1}\mathbf{B} \end{array}\right).$$

Erityisesti m_1-vektorin $\mathbf{a} = (a_1,\ldots,a_{m_1})$ ja m_2-vektorin **b** Kronecker-tulo on $m_1 m_2$-vektori

$$\mathbf{a} \otimes \mathbf{b} = \left(a_1\mathbf{b} \mid \cdots \mid a_{m_1}\mathbf{b}\right).$$

Kronecker-tulolla on seuraavat perusominaisuudet:

(1) Jos matriisitulot **AC** ja **BD** ovat määritellyt, niin

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}).$$

(Seuraa kutakuinkin suoraan lohkomatriisien kertolaskukaavasta.)

(2) $(\mathbf{A} \otimes \mathbf{B})^{\mathrm{T}} = \mathbf{A}^{\mathrm{T}} \otimes \mathbf{B}^{\mathrm{T}}$

(Seuraa kutakuinkin suoraan lohkomatriisien transponointikaavasta.)

(3) Jos matriisisummat $\mathbf{A} + \mathbf{C}$ ja $\mathbf{B} + \mathbf{D}$ ovat määritellyt, niin

$$(\mathbf{A} + \mathbf{C}) \otimes \mathbf{B} = \mathbf{A} \otimes \mathbf{B} + \mathbf{C} \otimes \mathbf{B}$$

ja

$$\mathbf{A} \otimes (\mathbf{B} + \mathbf{D}) = \mathbf{A} \otimes \mathbf{B} + \mathbf{A} \otimes \mathbf{D}.$$

(Seuraa kutakuinkin suoraan lohkomatriisien yhteenlaskukaavasta.)

(4) Jos c on skalaari, niin $c(\mathbf{A} \otimes \mathbf{B}) = (c\mathbf{A}) \otimes \mathbf{B} = \mathbf{A} \otimes (c\mathbf{B})$.

(5) $$\begin{pmatrix} \mathbf{A}_1 \\ \hline \vdots \\ \hline \mathbf{A}_j \end{pmatrix} \otimes \mathbf{B} = \begin{pmatrix} \mathbf{A}_1 \otimes \mathbf{B} \\ \hline \vdots \\ \hline \mathbf{A}_j \otimes \mathbf{B} \end{pmatrix}$$

(6) Yleisesti matriisit $\mathbf{A} \otimes \begin{pmatrix} \mathbf{B}_1 \\ \hline \vdots \\ \hline \mathbf{B}_j \end{pmatrix}$ ja $\begin{pmatrix} \mathbf{A} \otimes \mathbf{B}_1 \\ \hline \vdots \\ \hline \mathbf{A} \otimes \mathbf{B}_j \end{pmatrix}$ eivät ole samat, mutta niillä on samat rivit (vain eri järjestyksessä).

(7) Jos matriisin **A** rivit ovat lineaarisesti riippumattomat ja samoin matriisin **B**, niin myös matriisin $\mathbf{A} \otimes \mathbf{B}$ rivit ovat lineaarisesti riippumattomat.*

n_1n_2-vektori **c** voidaan kirjoittaa $n_1 \times n_2$-matriisiksi **C** jakamalla se n_2-pituisiin lohkoihin ja kirjoittamalla nämä n_1 lohkoa allekkain. Lineaaristen koodien C_1 ja C_2, joiden koodisymbolit ovat saman äärellisen kunnan alkioita, *tulokoodi* $C_1 \otimes C_2$ on kaikkien niiden vektorien **c** muodostama n_1n_2-pituinen lineaarinen koodi, joita vastaavien $n_1 \times n_2$-matriisien rivit ovat C_2:n koodivektoreita ja sarakkeet C_1:n koodivektoreita. Ilmeisesti $C_1 \otimes C_2$ on lineaarinen koodi.

LAUSE 15. Jos $d_{min}(C_1) = d_1$ ja $d_{min}(C_2) = d_2$, niin

$$d_{min}(C_1 \otimes C_2) = d_1d_2$$

ja näin ollen $C_1 \otimes C_2$ pystyy korjaamaan $\lfloor (d_1d_2 - 1)/2 \rfloor$ virhettä.

---

* Tämän tuloksen todistus on seuraavanlainen. Olkoot **A**:n rivit $\mathbf{a}_1,\ldots,\mathbf{a}_{n_1}$ ja **B**:n rivit $\mathbf{b}_1,\ldots,\mathbf{b}_{n_2}$. Kirjoitetaan

$$\mathbf{0}_{m_1m_2} = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} c_{ij}(\mathbf{a}_i \otimes \mathbf{b}_j) = \sum_{j=1}^{n_2} \underbrace{\left(\sum_{i=1}^{n_1} c_{ij}\mathbf{a}_i\right)}_{=\mathbf{d}_j=(d_{j1},\ldots,d_{jm_1})} \otimes \mathbf{b}_j$$

$$= \sum_{j=1}^{n_2} \left(d_{j1}\mathbf{b}_j \,|\cdots|\, d_{jm_1}\mathbf{b}_j\right) = \left(\sum_{j=1}^{n_2} d_{j1}\mathbf{b}_j \,\middle|\cdots\middle|\, \sum_{j=1}^{n_2} d_{jm_1}\mathbf{b}_j\right).$$

Mutta silloin

$$\sum_{j=1}^{n_2} d_{ji}\mathbf{b}_j = \mathbf{0}_{m_2} \quad (i = 1,\ldots,m_1)$$

ja, koska vektorit $\mathbf{b}_1,\ldots,\mathbf{b}_{n_2}$ ovat lineaarisesti riippumattomat, myös $\mathbf{d}_1 = \cdots = \mathbf{d}_{n_2} = \mathbf{0}_{m_1}$. Koska toisaalta vektorit $\mathbf{a}_1,\ldots,\mathbf{a}_{n_1}$ ovat lineaarisesti riippumattomat, niin näin ollen kaikki c_{ij}:t ovat $= \overline{0}$.

Todistus. Jos $\mathbf{c}_1 \in C_1$ ja $\mathbf{c}_2 \in C_2$, niin $\mathbf{c}_1 \otimes \mathbf{c}_2 \in C_1 \otimes C_2$ ja ilmeisesti $w(\mathbf{c}_1 \otimes \mathbf{c}_2) = w(\mathbf{c}_1)w(\mathbf{c}_2)$. Näin ollen $d_{min}(C_1 \otimes C_2) \le d_1 d_2$. Jos taas $\mathbf{c} \neq \mathbf{0}$ on $C_1 \otimes C_2$:n koodisana, niin $\mathbf{C}$:n $\mathbf{0}$:sta eroavien rivien painot ovat $\ge d_2$ ja näitä rivejä on ainakin d_1 kpl. Siispä $d_{min}(C_1 \otimes C_2) \ge d_1 d_2$. ■

LAUSE 16. Jos $\mathbf{G}_1$ ja $\mathbf{G}_2$ ovat C_1:n ja C_2:n generoijamatriiseja, niin $\mathbf{G}_1 \otimes \mathbf{G}_2$ on tulokoodin $C_1 \otimes C_2$ generoijamatriisi ja tulokoodin viestin pituus on $k_1 k_2$.

Todistus. $\mathbf{G}_1 \otimes \mathbf{G}_2$:n rivit ovat $C_1 \otimes C_2$:n koodisanoja, niitä on $k_1 k_2$ kpl ja ominaisuuden **(7)** nojalla ne ovat myös lineaarisesti riippumattomat. Edelleen, jos $\mathbf{m}_1\mathbf{G}_1 = \mathbf{c}_1$ ja $\mathbf{m}_2\mathbf{G}_2 = \mathbf{c}_2$, niin ominaisuuden **(1)** nojalla

$$(\mathbf{m}_1 \otimes \mathbf{m}_2)(\mathbf{G}_1 \otimes \mathbf{G}_2) = (\mathbf{m}_1\mathbf{G}_1) \otimes (\mathbf{m}_2\mathbf{G}_2) = \mathbf{c}_1 \otimes \mathbf{c}_2.$$

Vektorit $\mathbf{c}_1 \otimes \mathbf{c}_2$ ovat siis $\mathbf{G}_1 \otimes \mathbf{G}_2$:n rivien lineaariyhdelmiä. Toisaalta tulokoodin $C_1 \otimes C_2$ koodivektori $\mathbf{c}$ on lausuttavissa muotoa $\mathbf{c}_1 \otimes \mathbf{c}_2$ olevien vektoreiden lineaariyhdelmänä. Tämä nähdään etenemällä ketjua

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_1^{(0)} \\ \mathbf{c}_2^{(0)} \\ \mathbf{c}_3^{(0)} \\ \vdots \\ \mathbf{c}_{n_1}^{(0)} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{c}_2^{(1)} \\ \mathbf{c}_3^{(1)} \\ \vdots \\ \mathbf{c}_{n_1}^{(1)} \end{pmatrix} + \mathbf{c}_{11} \otimes \mathbf{c}_1^{(0)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{c}_3^{(2)} \\ \vdots \\ \mathbf{c}_{n_1}^{(2)} \end{pmatrix} + \mathbf{c}_{11} \otimes \mathbf{c}_1^{(0)} + \mathbf{c}_{21} \otimes \mathbf{c}_2^{(1)} = \cdots = \sum_{i=1}^{n_1} \mathbf{c}_{i1} \otimes \mathbf{c}_i^{(i-1)},$$

missä $\mathbf{c}_{11},\ldots,\mathbf{c}_{n_1 1} \in C_1$ valitaan sopivasti. (Tässä samaistetaan $\mathbf{c}_1 \otimes \mathbf{c}_2$ matriisimuotoonsa* $\mathbf{c}_1^T\mathbf{c}_2$.) ■

Jos $\mathbf{H}_1$ ja $\mathbf{H}_2$ ovat C_1:n ja C_2:n tarkistusmatriiseja, niin $\mathbf{H}_1 \otimes \mathbf{H}_2$ ei ole tulokoodin $C_1 \otimes C_2$ tarkistusmatriisi. Siinähän on jo liian vähän rivejäkin. $\mathbf{H}_1 \otimes \mathbf{H}_2$ voidaan kuitenkin rivejä lisäämällä täydentää tulokoodin tarkistusmatriisiksi:

LAUSE 17. Jos $\mathbf{H}_1$ ja $\mathbf{H}_2$ ovat C_1:n ja C_2:n tarkistusmatriiseja, niin matriisi

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \otimes \mathbf{H}_2 \\ \hline \mathbf{H}_1 \otimes \mathbf{A}_2 \\ \hline \mathbf{A}_1 \otimes \mathbf{H}_2 \end{pmatrix},$$

missä $k_1 \times n_1$-matriisi $\mathbf{A}_1$ ja $k_2 \times n_2$-matriisi $\mathbf{A}_2$ on valittu siten, että matriisien $\begin{pmatrix} \mathbf{H}_1 \\ \hline \mathbf{A}_1 \end{pmatrix}$ ja $\begin{pmatrix} \mathbf{H}_2 \\ \hline \mathbf{A}_2 \end{pmatrix}$ rivit ovat lineaarisesti riippumattomia, on tulokoodin $C_1 \otimes C_2$ generoijamatriisia $\mathbf{G} = \mathbf{G}_1 \otimes \mathbf{G}_2$ vastaava tarkistusmatriisi.

---

* Tämä on usein esiintyvä vektorioperaatio, ns. $\mathbf{c}_1$:n ja $\mathbf{c}_2$:n dyaditulo.

Todistus. Matriisissa $\mathbf{H}$ on $n_1n_2 - k_1k_2$ riviä. Ominaisuuksien **(5)** ja **(6)** nojalla $\mathbf{H}$:n rivit ovat ovat matriisin $\left(\frac{\mathbf{H}_1}{\mathbf{A}_1}\right) \otimes \left(\frac{\mathbf{H}_2}{\mathbf{A}_2}\right)$ rivien joukossa ja ominaisuuden **(7)** nojalla vm. matriisin rivit ovat lineaarisesti riippumattomat. Siispä myös $\mathbf{H}$:n rivit ovat lineaarisesti riippumattomat. Koska matriisit $\mathbf{G}_1\mathbf{H}_1^T$ ja $\mathbf{G}_2\mathbf{H}_2^T$ ovat nollamatriiseja, niin ominaisuuksien **(1)** ja **(2)** nojalla myös

$$\mathbf{G}\mathbf{H}^T = (\mathbf{G}_1 \otimes \mathbf{G}_2)\left(\frac{\mathbf{H}_1 \otimes \mathbf{H}_2}{\mathbf{H}_1 \otimes \mathbf{A}_2}{\mathbf{A}_1 \otimes \mathbf{H}_2}\right)^T = (\mathbf{G}_1 \otimes \mathbf{G}_2)\left(\mathbf{H}_1^T \otimes \mathbf{H}_2^T \mid \mathbf{H}_1^T \otimes \mathbf{A}_2^T \mid \mathbf{A}_1^T \otimes \mathbf{H}_2^T\right)$$
$$= \left(\left(\mathbf{G}_1\mathbf{H}_1^T\right) \otimes \left(\mathbf{G}_2\mathbf{H}_2^T\right) \mid \left(\mathbf{G}_1\mathbf{H}_1^T\right) \otimes \left(\mathbf{G}_2\mathbf{A}_2^T\right) \mid \left(\mathbf{G}_1\mathbf{A}_1^T\right) \otimes \left(\mathbf{G}_2\mathbf{H}_2^T\right)\right)$$

on nollamatriisi. ■

Erityisen yksinkertainen asia on, jos kyseessä on systemaattinen koodaus. Silloin nimittäin

$$\mathbf{G}_1 = \left(\mathbf{I}_{k_1} \mid \mathbf{P}_1\right), \quad \mathbf{G}_2 = \left(\mathbf{I}_{k_2} \mid \mathbf{P}_2\right)$$

ja (Lause **3**)

$$\mathbf{H}_1 = \left(-\mathbf{P}_1^T \mid \mathbf{I}_{n_1-k_1}\right), \quad \mathbf{H}_2 = \left(-\mathbf{P}_2^T \mid \mathbf{I}_{n_2-k_2}\right)$$

ja voidaan valita

$$\mathbf{A}_1 = \left(\mathbf{I}_{k_1} \mid \mathbf{O}_{k_1,n_1-k_1}\right), \quad \mathbf{A}_2 = \left(\mathbf{I}_{k_2} \mid \mathbf{O}_{k_2,n_2-k_2}\right).$$

Koodien tuloa muodostettaessa paranee purskevirheen korjauskyky enemmän kuin virheenkorjauskyky:

LAUSE 18. Jos tulokoodissa $C_1 \otimes C_2$ koodi C_1 korjaa enintään d-pituiset purskevirheet ja C_2 on n_2-pituinen, niin $C_1 \otimes C_2$ korjaa enintään dn_2-pituiset purskevirheet.

Todistus. Tässä yhteydessä kannattaa tulokoodin koodisana $\mathbf{c}$ kirjoittaa $n_1 \times n_2$-matriisiksi $\mathbf{C}$. Silloin $\mathbf{C}$:n sarakkeet ovat C_1:n koodisanoja ja enintään dn_2-pituinen purskevirhe $\mathbf{c}$:ssä aiheuttaa ko. sarakkeisiin enintään d-pituiset purskevirheet, jotka voidaan korjata. ■

Tulokoodit ovat näin erityisen otollisia purskevirheiden korjauksen kannalta ja lauseen todistus antaa dekoodausmenetelmän.

ESIMERKKI. Binäärinen (rk,k)-toistokoodi C saadaan (r,1)-toistokoodin $C_1 = \{0^r, 1^r\}$ sekä triviaalin koodin $C_2 = \{0,1\}^k$ (jossa koodisana on sama kuin viestisana) tulokoodina. C_1 pystyy korjaamaan enintään $\lfloor (r-1)/2 \rfloor$–pituiset purskevirheet, joten tulokoodi C korjaa enintään $\lfloor (r-1)/2 \rfloor$k-pituiset purskevirheet (vrt. Esimerkki s. 43).

Sellaisen dekoodausmenetelmän kehittäminen, joka tehokkaasti korjaisi tavalliset "hajavirheet" aina maksimaaliseen määrään $\lfloor (d_1 d_2 - 1)/2 \rfloor$ (Lause **15**) asti, onkin jo pulmallisempaa. Mikäli käytetään hyväksi koodin C_1 pyyhkiymien korjauskykyä, saadaan seuraava menettely. Muistettaneen, että C_1 pystyy korjaamaan yhtaikaisesti u virhettä ja w pyyhkiymää tarkalleen siinä tapauksessa, että $2u + w + 1 \leq d_1$ (ks. s. 3). Erityisen hyvä tällainen pyyhkiymien ja "tavallisten" virheiden yhteiskorjauskyky on BCH- ja Reed–Solomon-koodeilla, ks. **III.5**. Menettelyä varten kirjoitetaan saatu vektori **r** matriisimuotoon **R** vastaten tulokoodin koodisanan **c** matriisimuotoa **C**.

1) Dekoodataan **R**:n rivit C_2:n dekoodausalgoritmilla. Rivit, joita ei voida dekoodata, pyyhitään pois, ts. niiden symbolit korvataan pyyhkiymillä. Jokaista (i:ttä) riviä kohti talletetaan korjausten lukumäärä κ_i. Näin saadaan matriisi **R′**. (**R′**:ssa on mukana pyyhkiymät jollakin tavoin merkittyinä.)

2) Lasketaan tulokoodin virheenkorjauskyky

$$t = \left\lfloor \frac{(d_1 - e)d_2 - 1}{2} \right\rfloor,$$

missä e on pois pyyhittyjen rivien lukumäärä.

3) Dekoodataan **R′**:n sarakkeet ensimmäisestä lähtien yksi kerrallaan käyttäen pyyhkiymiä ja "tavallisia" virheitä yhtaikaa korjaavaa C_1:n dekoodausalgoritmia. Jos dekoodaus onnistuu, lasketaan virheluku ν seuraavasti:

a) Asetetaan $\nu \leftarrow 0$ ja $i \leftarrow 1$.

b) Jos i:ttä riviä ei ole jo pyyhitty pois, asetetaan $\nu \leftarrow \nu + \kappa_i$, mikäli käsillä olevan sarakkeen i:ttä alkiota ei korjattu sarakkeen dekoodauksessa, ja $\nu \leftarrow \nu + d_2 - \kappa_i$, mikäli i:s alkio korjattiin. (Vm. tapauksessa, jos sarake dekoodautui oikein, niin i:s rivi korjattiin väärään C_2:n koodisanaan ja i:nnellä rivillä oli alunperin ainakin $d_2 - \kappa_i$ virhettä.)

c) Jos $i = n_1$, lopetetaan. Muussa tapauksessa asetetaan $i \leftarrow i + 1$ ja mennään **b)**:hen.

Jos dekoodaus ei onnistu tai onnistui, mutta $\nu > t$, niin pyyhitään jälleen pois sellainen rivi, jolla on suurin κ_i, otetaan uudeksi $\mathbf{R}'$:ksi näin saatu matriisi, ja mennään **2)**:een. Muussa tapauksessa sarakkeen dekoodaus onnistui ja $\nu \leq t$ ja siirrytään seuraavaan sarakkeeseen.

4) Menettely pysähtyy, kun kaikki sarakkeet on saatu dekoodatuksi tai kun kaikki rivit on pyyhitty pois. Ensin mainitussa tapauksessa $\mathbf{r}$:n dekoodaus onnistui, viimemainitussa tapauksessa taas virhe paljastui, mutta sitä ei voitu korjata.

Tulokoodeja sekä niiden dekoodausta on käsitelty laajemmin viitteissä BLAHUT ja LIN & COSTELLO, ks. myös SWEENEY.

3. Kiedotut koodit

Koodista C saadaan *ℓ-kertaisesti kiedottu koodi* $C^{(\ell)}$ seuraavasti. Otetaan kaikin tavoin ℓ C:n koodisanaa $\mathbf{c}_i = c_{i1}c_{i2}\cdots c_{in}$ $(i = 1,\ldots,\ell)$ (eivät välttämättä kaikki eri sanoja) ja muodostetaan niistä matriisi

$$\begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_\ell \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{\ell 1} & c_{\ell 2} & \cdots & c_{\ell n} \end{pmatrix}.$$

Vastaava $C^{(\ell)}$:n koodisana saadaan lukemalla matriisi sarakkeittain ylhäältä alaspäin. Jos C on (n,k)-koodi, niin $C^{(\ell)}$ on $(\ell n, \ell k)$-koodi. Polynomimuodossa yo. matriisia vastaa polynomi

$$\mathbf{c}_1(z^\ell) + z\mathbf{c}_2(z^\ell) + z^2\mathbf{c}_3(z^\ell) + \cdots + z^{\ell-1}\mathbf{c}_\ell(z^\ell).$$

LAUSE 19. Jos C korjaa enintään d-pituiset purskevirheet, niin $C^{(\ell)}$ korjaa enintään ℓd-pituiset purskevirheet. Näin ollen, jos C on Reiger-optimaalinen, niin samoin on $C^{(\ell)}$.

Todistus. Enintään ℓd-pituinen purskevirhe $C^{(\ell)}$:n koodisanassa aiheuttaa enintään d-pituiset purskevirheet koodisanan konstruktiossa käytetyissä C:n koodisanoissa $\mathbf{c}_1,\ldots,\mathbf{c}_\ell$. ■

Lineaarisuus säilyy kietomisessa ja samoin syklisyys:

LAUSE 20. Jos C on syklinen (n,k)-koodi, jonka generoijapolynomi on $\mathbf{g}(z)$ ja vastaava tarkistuspolynomi $\mathbf{h}(z)$, niin $C^{(\ell)}$ on syklinen $(\ell n, \ell k)$-koodi, jonka generoijapolynomi on $\mathbf{g}(z^\ell)$ ja vastaava tarkistuspolynomi $\mathbf{h}(z^\ell)$.

Todistus. Jos $C^{(\ell)}$:n sanaa vastaava matriisi on

$$\begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_\ell \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{\ell 1} & c_{\ell 2} & \cdots & c_{\ell n} \end{pmatrix},$$

niin syklisellä yhden symbolin siirrolla saatavaa sanaa vastaa matriisi

$$\begin{pmatrix} c_{21} & c_{22} & \cdots & c_{2n} \\ c_{31} & c_{32} & \cdots & c_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{12} & c_{13} & \cdots & c_{11} \end{pmatrix},$$

jonka rivit $\mathbf{c}_2,\ldots,\mathbf{c}_\ell$ ja $c_{12}c_{13}\cdots c_{1,n-1}c_{11}$ ovat C:n koodisanoja. Jos siis C on syklinen, niin samoin on $C^{(\ell)}$.

Oletetaan sitten, että $\mathbf{g}(z)$ on C:n generoijapolynomi ja $\mathbf{h}(z)$ vastaava tarkistuspolynomi. Koska $\mathbf{g}(z)$ on $z^n - \overline{1}$:n tekijä, niin $\mathbf{g}(z^\ell)$ on $z^{n\ell} - \overline{1}$:n tekijä. Otetaan tarkasteltavaksi (mielivaltainen) $C^{(\ell)}$:n viestipolynomi $\mathbf{m}(z)$ ja kirjoitetaan se muotoon

$$\mathbf{m}(z) = \mathbf{m}_1(z^\ell) + z\mathbf{m}_2(z^\ell) + \cdots + z^{\ell-1}\mathbf{m}_\ell(z^\ell).$$

Silloin polynomit $\mathbf{m}_1(z),\ldots,\mathbf{m}_\ell(z)$ ovat (mielivaltaisia) C:n viestipolynomeja, niitä vastaavat C:n koodipolynomit $\mathbf{c}_i(z) = \mathbf{m}_i(z)\mathbf{g}(z)$ $(i = 1,\ldots,\ell)$ ja

$$\begin{aligned} \mathbf{m}(z)\mathbf{g}(z^\ell) &= \Big(\mathbf{m}_1(z^\ell) + z\mathbf{m}_2(z^\ell) + \cdots + z^{\ell-1}\mathbf{m}_\ell(z^\ell)\Big)\mathbf{g}(z^\ell) \\ &= \mathbf{c}_1(z^\ell) + z\mathbf{c}_2(z^\ell) + z^2\mathbf{c}_3(z^\ell) + \cdots + z^{\ell-1}\mathbf{c}_\ell(z^\ell). \end{aligned}$$

$C^{(\ell)}$:n koodipolynomit ovat näin tarkalleen kaikki muotoa $\mathbf{m}(z)\mathbf{g}(z^\ell)$ olevat polynomit. Koska $z^{n\ell} - \overline{1} = \mathbf{g}(z^\ell)\mathbf{h}(z^\ell)$, on $\mathbf{h}(z^\ell)$ $C^{(\ell)}$:n $\mathbf{g}(z^\ell)$:tä vastaava tarkistuspolynomi. ■

Kietomalla voidaan siis esimerkiksi muodostaa mielivaltaisen pitkiä purskevirheitä korjaavia Reiger-optimaalisia syklisiä koodeja käyttäen s. 45 olevan taulukon koodeja.

Koodisanojen $\mathbf{c}_1,\ldots,\mathbf{c}_\ell$ avulla saatu matriisi voidaan lukea muillakin tavoin kiedotuksi koodisanaksi purskevirheenkorjauskyvyn pysyessä sykliselle koodille C Lauseen **19** mukaisena (yo. tapa on ns. *lohkokietominen*). Voi-

daan edetä esimerkiksi lävistäjittäin vasemmalta oikealle ja ylhäältä alas (lihavointi) aloittaen päälävistäjästä

$$\begin{array}{ccccc|c} \mathbf{c_{11}} & \mathbf{c_{12}} & \mathbf{c_{13}} & \mathbf{c_{14}} & \mathbf{c_{15}} & \\ c_{21} & \mathbf{c_{22}} & \mathbf{c_{23}} & \mathbf{c_{24}} & \mathbf{c_{25}} & \mathbf{c_{21}} \\ c_{31} & c_{32} & \mathbf{c_{33}} & \mathbf{c_{34}} & \mathbf{c_{35}} & \mathbf{c_{31}}\ \mathbf{c_{32}} \\ c_{41} & c_{42} & c_{43} & \mathbf{c_{44}} & \mathbf{c_{45}} & \mathbf{c_{41}}\ \mathbf{c_{42}}\ \mathbf{c_{43}} \end{array}$$

tai vastalävistäjästä

$$\begin{array}{ccccc|c} c_{11} & c_{12} & c_{13} & \mathbf{c_{14}} & \mathbf{c_{15}} & \mathbf{c_{11}}\ \mathbf{c_{12}}\ \mathbf{c_{13}} \\ c_{21} & c_{22} & \mathbf{c_{23}} & \mathbf{c_{24}} & \mathbf{c_{25}} & \mathbf{c_{21}}\ \mathbf{c_{22}} \\ c_{31} & \mathbf{c_{32}} & \mathbf{c_{33}} & \mathbf{c_{34}} & \mathbf{c_{35}} & \mathbf{c_{31}} \\ \mathbf{c_{41}} & \mathbf{c_{42}} & \mathbf{c_{43}} & \mathbf{c_{44}} & \mathbf{c_{45}} & \end{array}$$

Jälkimmäisen tavan muunnelma, jossa $\ell = n$ ja sovitaan, että ensimmäiset ja viimeiset $\ell - 1$ C:n koodisanaa ovat kiinteitä (vaikkapa $\mathbf{c} = c_1 \cdots c_n$), on erityisen kätevä kiedottaessa uusia C:n koodisanoja jatkuvasti mukaan "unohtaen" entisiä, ks. oheinen esimerkki. Lävistäjät luetaan koodisanaksi ylimmästä aloittaen alhaalta ylös (miksei ylhäältä alas?). Tämä menettely on ns. *viivekiedonta*.

$$\begin{array}{cccc} c_1 & c_2 & c_3 & \mathbf{c_4} \\ c_1 & c_2 & \mathbf{c_3} & \mathbf{c_4} \\ c_1 & \mathbf{c_2} & \mathbf{c_3} & \mathbf{c_4} \\ \mathbf{c_{41}} & \mathbf{c_{42}} & \mathbf{c_{43}} & \mathbf{c_{44}} \\ \mathbf{c_{51}} & \mathbf{c_{52}} & \mathbf{c_{53}} & \mathbf{c_{54}} \\ \mathbf{c_{61}} & \mathbf{c_{62}} & \mathbf{c_{63}} & \mathbf{c_{64}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{c_{N1}} & \mathbf{c_{N2}} & \mathbf{c_{N3}} & \mathbf{c_{N4}} \\ \mathbf{c_1} & \mathbf{c_2} & \mathbf{c_3} & c_4 \\ \mathbf{c_1} & \mathbf{c_2} & c_3 & c_4 \\ \mathbf{c_1} & c_2 & c_3 & c_4 \end{array}$$

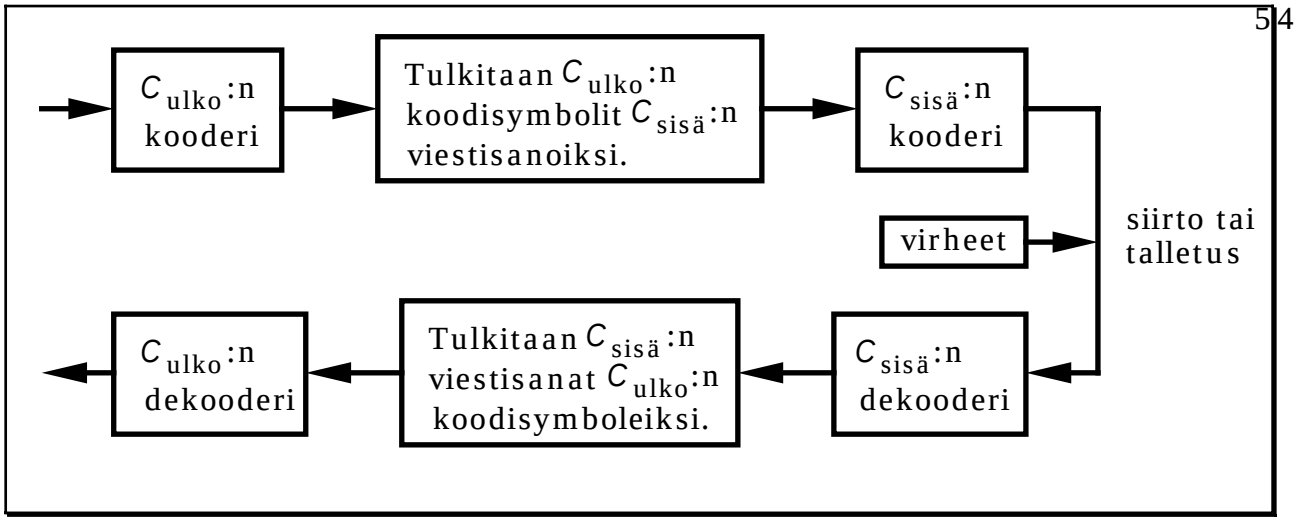
Kietominen voidaan tehdä myös käyttäen useampaa koodia. *Ristikietomisessa* tehdään (k_1,n_1)-koodista C_1 ja (k_2,n_2)-koodista C_2 kiedottu koodi seuraavasti. Muodostetaan k_2:sta C_1:n koodisanasta $\mathbf{c}_1,\ldots,\mathbf{c}_k$ matriisi

$$\begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_{k_2} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n_1} \\ c_{21} & c_{22} & \cdots & c_{2n_1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k_2 1} & c_{k_2 2} & \cdots & c_{k_2 n_1} \end{pmatrix}$$

aivan kuten edellä. Sen sijaan, että luettaisiin matriisi sarakkeittain koodisanaksi, tulkitaankin nyt sarakkeet C_2:n viestisanoiksi ja koodataan ne. Näin saadaan n_1 kpl C_2:n koodisanoja, jotka taas kiedotaan joko kaikki tai sitten ℓ:n ryhmissä (jos ℓ ei ole n_1:n tekijä, voidaan yli jääneet C_2:n koodisanat "säästää" ja käyttää seuraavassa kietomisessa). Ristikiedotun koodin ominaisuudet ovat paljolti samantapaiset kuin katenoidun koodin (ks. seuraava pykälä).

4. Katenoidut koodit

Koodien *katenoinnissa* ideana on koodata ensin yhdellä koodilla C_{ulko}, ns. *ulkokoodilla*, tulkita koodisymbolit toisen koodin $C_{\text{sisä}}$, ns. *sisäkoodin*, viestisanoiksi sekä sitten koodata vielä tällä koodilla. Dekoodattaessa toimitaan päinvastaisessa järjestyksessä.



Vastaavalla tavalla voitaisiin katenoida useampiakin kuin kaksi koodia.

Yleensä käytetään ulkokoodina ja sisäkoodina lineaarisia koodeja, joiden koodisymbolit ovat vastaavasti äärellisten kuntien $GF(p^M)$ ja $GF(p^m)$ alkioita, missä m on M:n tekijä eli M = km. Tällöin $GF(p^M)$:n alkiot voidaan samaistaa M-pituisiksi vektoreiksi, joiden komponentit ovat alkukunnan $\mathbb{Z}_p$ alkioita. Koska m on M:n tekijä, voidaan nämä vektorit lohkoa m-pituisiin osavektoreihin, jotka taas voidaan edelleen samaistaa $GF(p^m)$:n alkioihin. Näin jokainen ulkokoodin koodisymboli voidaan samaistaa k-pituiseksi vektoriksi, jonka alkiot ovat sisäkoodin koodisymboleja. Jos nyt C_{ulko} on (N,K)-koodi ja $C_{sisä}$ on (n,k)-koodi ja tulkitaan myös C_{ulko}:n viestisymbolit k-pituisiksi vektoreiksi, joiden alkiot ovat sisäkoodin koodisymboleja, niin katenoimalla saatu koodi on lineaarinen (Nn,Kk)-koodi.

Katenoidussa koodissa sisäkoodin tehtävänä on korjata “hajavirheet”. Ulkokoodi puolestaan korjaa purskevirheet, jotka sen näkökulmasta ovat tavallisia hajavirheitä. Jos ulkokoodilla on vielä hyvä purskevirheiden korjauskyky, voidaan katenoidulla koodilla korjata pitkiäkin purskevirheitä. Purskevirheen sattuessa sisäkoodi joko dekoodaa väärin, mikä aiheuttaa ulkokoodille yhden virheen, tai sitten ei pysty dekoodaamaan lainkaan. Jälkimmäisessä tapauksessa voidaan joko ilmoittaa ulkokoodille vastaava koodisymboli pyyhkiytyneeksi — ja käyttää ulkokoodin mahdollista tehokasta pyyhkiymien korjausta — tai sitten antaa ulkokoodille mielivaltainen koodisymboli, joka todennäköisesti on virheellinen.

Hyvän katenoimalla saadun koodin suunnittelu ei ole aivan helppo tehtävä.* Sisäkoodin viestin pituuden k tulisi olla pieni, jotta ulkokoodin koodisymbolit eivät tulisi kovin suuresta kunnasta $GF(p^{km})$. Tämä taas joko rajoittaa sisäkoodin tehokkuutta (n on liian suuri k:hon verrattuna) tai sitten sen virheenkorjauskykyä ja ulkokoodi saakin korjattavakseen paljon virheitä. Ks. SWEENEY.

---

* Alkuperäisviite FORNEY, G.D.: *Concatenated Codes*. MIT Press (–66) on edelleen hyvin käyttökelpoinen.

5. Fire-koodit

Otetaan koodisymbolit alkukunnasta $\mathbb{Z}_p$ ja valitaan generoijapolynomiksi

$$\mathbf{g}(z) = (z^c - \overline{1})\mathbf{p}(z),$$

missä $\mathbf{p}(z)$ on astetta m oleva $\mathbb{Z}_p[z]$:n jaoton polynomi ja c valitaan sopivasti. (Yleisemminkin voitaisiin valita koodisymbolit mielivaltaisesta äärellisestä kunnasta.) Polynomin $\mathbf{p}(z)$ ns. *jakso* on pienin sellainen luku e, että $\mathbf{p}(z)$ on polynomin $z^e - \overline{1}$ tekijä. (Koska $\mathbf{p}(z)$:tä voitaisiin käyttää äärellisen kunnan GF(p^m) konstruktiossa, se on minimipolynomi, joten sillä on jakso, ks. **III.1**.) Nyt c valitaan siten, että e ei ole sen tekijä. Koodin pituudeksi n otetaan c:n ja e:n pienin yhteinen jaettava pyj(c,e) eli pienin luku, jonka sekä c että e jakavat tasan.

Jotta saataisiin syklinen koodi, pitää $\mathbf{g}(z)$:n olla polynomin $z^n - \overline{1}$ tekijä. Tämän toteaminen* ei ole ihan helppoa.

Fire-koodeilla on erinomaisen hyvät purskevirheiden paljastus- ja korjaamisominaisuudet, jopa yhtaikaisesti. Asian selvittämiseksi esitetään seuraava dekoodausmenettely.

(1) Saatu polynomi on muotoa $\mathbf{r}(z) = \mathbf{c}(z) + z^i\mathbf{b}(z)$, missä $\mathbf{c}(z)$ on koodipolynomi ja $z^i\mathbf{b}(z)$ on purskevirhepolynomi ja $\mathbf{b}(z)$:n vakiotermi on $\neq \overline{0}$. Muodostetaan siitä jakamalla $z^c - \overline{1}$:llä ja $\mathbf{p}(z)$:lla *purskearvosyndromi* $\mathbf{s}_1(z)$ sekä *purskekohtasyndromi* $\mathbf{s}_2(z)$:

$$z^i\mathbf{b}(z) = \mathbf{q}_1(z)(z^c - \overline{1}) + \mathbf{s}_1(z), \qquad \deg(\mathbf{s}_1(z)) < c,$$

$$z^i\mathbf{b}(z) = \mathbf{q}_2(z)\mathbf{p}(z) + \mathbf{s}_2(z), \qquad \deg(\mathbf{s}_2(z)) < m.$$

---

* Todetaan ensin yleisesti, että $z^u - \overline{1}$ on $z^v - \overline{1}$:n tekijä siinä tapauksessa, että u on v:n tekijä. Jos nimittäin u on v:n tekijä, niin v = uw. Ilmeisesti $z - \overline{1}$ on $z^w - \overline{1}$:n tekijä, joten tällöin edelleen $z^u - \overline{1}$ on $z^{uw} - \overline{1}$:n eli $z^v - \overline{1}$:n tekijä. Nyt c on n:n tekijä, joten $z^c - \overline{1}$ on $z^n - \overline{1}$:n tekijä. Toisaalta $\mathbf{p}(z)$ on $z^e - \overline{1}$:n tekijä ja e on n:n tekijä, joten myös $\mathbf{p}(z)$ on $z^n - \overline{1}$:n tekijä. Edelleen c ei ole e:n tekijä, joten jaettaessa c e:llä saadaan c = qe + j, missä $0 < j < e$. Jos nyt $\mathbf{p}(z)$ olisi $z^c - \overline{1}$:n tekijä, olisi se myös erotuksen $(z^c - \overline{1}) - (z^e - \overline{1}) = z^e(z^{c-e} - \overline{1})$ tekijä ja jaottomana $z^{c-e} - \overline{1}$:n tekijä. Jatkamalla näin q kertaa todettaisiin lopulta, että $\mathbf{p}(z)$ olisi $z^j - \overline{1}$:n tekijä, mikä on mahdotonta, koska $0 < j < e$. Siispä $\mathbf{p}(z)$ ei ole $z^c - \overline{1}$:n tekijä ja, koska $\mathbf{p}(z)$ on jaoton, $\overline{1} = \mathrm{syt}(\mathbf{p}(z), z^c - \overline{1})$ ja (ks. **III.1**) $\overline{1} = \mathbf{a}(z)\mathbf{p}(z) + \mathbf{b}(z)(z^c - \overline{1})$. Kuten todettiin, $z^c - \overline{1}$ on $z^n - \overline{1}$:n tekijä, joten $z^n - \overline{1} = \mathbf{f}(z)(z^c - \overline{1})$. Koska myös $\mathbf{p}(z)$ on $z^n - \overline{1}$:n tekijä, on se edelleen polynomin $\mathbf{f}(z) = \mathbf{f}(z)\mathbf{a}(z)\mathbf{p}(z) + \mathbf{b}(z)\mathbf{f}(z)(z^c - \overline{1}) = \mathbf{f}(z)\mathbf{a}(z)\mathbf{p}(z) + \mathbf{b}(z)(z^n - \overline{1})$ tekijä. Kaiken kaikkiaan siis $\mathbf{g}(z) = (z^c - \overline{1})\mathbf{p}(z)$ on $z^n - \overline{1}$:n tekijä, kuten pitikin.

(2) Jos $\mathbf{s}_1(z) = \mathbf{s}_2(z) = \overline{0}$, katsotaan $\mathbf{r}(z)$ virheettömäksi* ja lopetetaan.

(3) Jos nyt purskevirheen pituus on enintään c eli $\deg(\mathbf{b}(z)) \leq c - 1$, on $\mathbf{s}_1(z) \neq \overline{0}$ ja purskevirhe paljastuu (se voidaan vielä kuitenkin korjata väärin!). Jos samalla $\mathbf{s}_2(z) = \overline{0}$, katsotaan purskevirhe vain paljastuneeksi ja lopetetaan. Pitkä purskevirhe voi myös johtaa tilanteeseen, jossa $\mathbf{s}_1(z) = \overline{0}$ ja $\mathbf{s}_2(z) \neq \overline{0}$, ja myös tällöin katsotaan se vain paljastuneeksi ja lopetetaan.

(4) Muussa tapauksessa sekä $\mathbf{s}_1(z) \neq \overline{0}$ että $\mathbf{s}_2(z) \neq \overline{0}$. Tällöin yritetään korjata purskevirhe. Sitä varten lasketaan siirretyt purskearvo- ja purskekohtasyndromit $\mathbf{s}_1^{(\ell)}(z)$ ja $\mathbf{s}_2^{(\ell)}(z)$ jakamalla

$$z^\ell \mathbf{s}_1(z) = \mathbf{q}_1^{(\ell)}(z)(z^c - \overline{1}) + \mathbf{s}_1^{(\ell)}(z), \qquad \deg\left(\mathbf{s}_1^{(\ell)}(z)\right) < c,$$
$$z^\ell \mathbf{s}_2(z) = \mathbf{q}_2^{(\ell)}(z)\mathbf{p}(z) + \mathbf{s}_2^{(\ell)}(z), \qquad \deg\left(\mathbf{s}_2^{(\ell)}(z)\right) < m$$

peräkkäin arvoille $\ell = 0,1,2,\ldots$. (Huomaa, että $\mathbf{s}_1^{(\ell)}(z)$ itse asiassa saadaan syklisellä siirrolla $\mathbf{s}_1(z)$:stä ja että $\mathbf{s}_1^{(0)}(z) = \mathbf{s}_1(z)$ sekä $\mathbf{s}_2^{(0)}(z) = \mathbf{s}_2(z)$.)

(5) Kun löytyy tilanne, missä nämä syndromit $\mathbf{s}_1^{(\ell)}(z)$ ja $\mathbf{s}_2^{(\ell)}(z)$ ovat samat, löytyy samalla i sekä $\mathbf{b}(z)$ ja purskevirhe voidaan korjata, edellyttäen että sen pituus on $t \leq m$. Mainittu tilanne tulee nimittäin tällöin lopulta vastaan, sillä

$$\mathbf{s}_1^{(n-i)}(z) = \mathbf{s}_2^{(n-i)}(z) = \mathbf{b}(z).$$

Jos nyt $c \geq s = \deg(\mathbf{b}(z)) + 1 \geq t$ ja $\mathbf{b}$:ssä on ainakin $c - t$ peräkkäistä nolla-alkiota, voi paljastunut s-pituinen purskevirhe tulla näin väärin dekoodatuksi, mutta muutoin sitä ei tapahdu. Näin ollen tällainen purskevirhe paljastuukin varmasti (ts. tulematta väärin dekoodatuksi), jos sen pituus on $s \leq c - t + 1$.

(6) Ellei **(5)**:ssä mainittua tilannetta esiinny $n - 1$ siirron aikana, on purskevirhe liian pitkä korjattavaksi. Sen katsotaan vain paljastuneen ja lopetetaan.

---

* Tällöin nimittäin sekä $z^c - \overline{1}$ että $\mathbf{p}(z)$ ovat $\mathbf{r}(z)$:n tekijöitä. Kirjoitetaan $\mathbf{r}(z) = \mathbf{v}(z)(z^c - \overline{1})$. Koska $\overline{1} = \mathrm{syt}(\mathbf{p}(z), z^c - \overline{1})$ (ks. edellinen alaviite), voidaan kirjoittaa $\overline{1} = \mathbf{a}_1(z)\mathbf{p}(z) + \mathbf{a}_2(z)(z^c - \overline{1})$ ja edelleen $\mathbf{v}(z) = \mathbf{v}(z)\mathbf{a}_1(z)\mathbf{p}(z) + \mathbf{a}_2(z)\mathbf{r}(z)$. $\mathbf{p}(z)$ siis on $\mathbf{v}(z)$:n tekijä ja näin ollen $\mathbf{g}(z) = (z^c - \overline{1})\mathbf{p}(z)$ on $\mathbf{r}(z)$:n tekijä eli $\mathbf{r}(z)$ on todellakin koodipolynomi.

HUOM! Algoritmia tutkiessa voi arvella, että ellei korjaamista yritetä, niin sillä saavutetaan osin vieläkin parempi purskevirheen paljastuskyky. Itse asiassa voidaan esimerkiksi näyttää, että s_1-pituisen ja s_2-pituisen purskevirheen muodostama purskevirhepari paljastuu, mikäli $m \geq \min(s_1,s_2)$ ja $s_1 + s_2 \leq c + 1$. (Tämä ei ole ihan helppoa, jossei nyt niin kovin vaikeaakaan.)

Esitetyssä dekoodausmenettelyssä voi siirrettyjä syndromeja joutua laskemaan enimmillään $ec - 1$ kpl. Nopeampi menettely saadaan huomaamalla, että koska $\mathbf{p}(z)$ on $z^e - \overline{1}$:n tekijä, niin voidaan kirjoittaa $z^e = \overline{1} + \mathbf{o}(z)\mathbf{p}(z)$ jollekin polynomille $\mathbf{o}(z)$. Näin ollen siirrettyjen syndromien jono $\mathbf{s}_2^{(0)}(z)$, $\mathbf{s}_2^{(1)}(z)$,... toistuu jaksollisesti samana $\mathbf{s}_2^{(e-1)}(z)$:n jälkeen*. Tämän nopeamman dekoodausmenettelyn kohdat **(1)–(3)** ovat samat kuin edellä ja loput kohdat ovat seuraavat:

(4') Muussa tapauksessa sekä $\mathbf{s}_1(z) \neq \overline{0}$ että $\mathbf{s}_2(z) \neq \overline{0}$. Tällöin yritetään korjata purskevirhe. Sitä varten lasketaan siirrettyjä purskearvosyndromeja $\mathbf{s}_1^{(\ell)}(z)$ jakamalla

$$z^\ell \mathbf{s}_1(z) = \mathbf{q}_1^{(\ell)}(z)(z^c - \overline{1}) + \mathbf{s}_1^{(\ell)}(z), \qquad \deg\left(\mathbf{s}_1^{(\ell)}(z)\right) < c$$

peräkkäin arvoille $\ell = 0,1,2,\ldots$, kunnes $\deg\left(\mathbf{s}_1^{(\ell)}(z)\right) < t$ ja talletetaan tarvittavien siirtojen määrä $\ell = \ell_1$ sekä silloinen syndromi $\mathbf{s}_1^{(\ell_1)}(z)$. (Huomaa, että itse asiassa $\mathbf{s}_1^{(\ell)}(z)$ saadaan syklisellä siirrolla $\mathbf{s}_1(z)$:stä ja että $\mathbf{s}_1^{(0)}(z) = \mathbf{s}_1(z)$.) Jollei tämä onnistu enintään $c - 1$ siirrolla, on purskevirhe liian pitkä korjattavaksi, jolloin se katsotaan paljastuneeksi ja lopetetaan.

(5') Lasketaan siirrettyjä purskekohtasyndromeja $\mathbf{s}_2^{(\ell)}(z)$ jakamalla

$$z^\ell \mathbf{s}_2(z) = \mathbf{q}_2^{(\ell)}(z)\mathbf{p}(z) + \mathbf{s}_2^{(\ell)}(z), \qquad \deg\left(\mathbf{s}_2^{(\ell)}(z)\right) < m$$

peräkkäin arvoille $\ell = 0,1,2,\ldots$, kunnes $\mathbf{s}_2^{(\ell)}(z) = \mathbf{s}_1^{(\ell_1)}(z)$ ja talletetaan tarvittavien siirtojen määrä $\ell = \ell_2$. Jollei tämä onnistu enintään $e - 1$ siirrolla, on purskevirhe liian pitkä korjattavaksi, jolloin se katsotaan paljastuneeksi ja lopetetaan.

---

* Mutta ei sitä ennen! Muutoin nimittäin $\mathbf{p}(z)$ on $\mathbf{b}(z)(z^d - \overline{1})$:n tekijä ja $0 < d < e$. Koska $\mathbf{p}(z)$ ei ole $\mathbf{b}(z)$:n tekijä, on $\overline{1} = \text{syt}(\mathbf{p}(z),\mathbf{b}(z))$ ja $\overline{1} = \mathbf{a}_1(z)\mathbf{p}(z) + \mathbf{a}_2(z)\mathbf{b}(z)$ ja $z^d - \overline{1} = (z^d - \overline{1})\mathbf{a}_1(z)\mathbf{p}(z) + (z^d - \overline{1})\mathbf{a}_2(z)\mathbf{b}(z)$ ja $\mathbf{p}(z)$ on silloin $z^d - \overline{1}$:n tekijä. Tämä on kuitenkin mahdotonta, sillä $d < e$ ja e on $\mathbf{p}(z)$:n jakso.

(6') Siirtojen määristä ℓ_1 sekä ℓ_2 saadaan nyt laskettua kohdassa **(5)** (edellinen algoritmi) tarvittava siirtojen määrä ℓ. Periodisuuksista johtuen nimittäin ℓ toteuttaa jotkin yhtälöt

$$\begin{cases} \ell = a_1 c + \ell_1 \\ \ell = a_2 e + \ell_2, \end{cases}$$

missä a_1 ja a_2 ovat kokonaislukuja. Jos nyt olisi kaksi ratkaisua $\ell = \ell'$ ja $\ell = \ell''$, missä $\ell' > \ell''$, niin saataisiin yhtälöt

$$\begin{cases} \ell' = a_1' c + \ell_1 \\ \ell' = a_2' e + \ell_2 \end{cases} \quad \text{ja} \quad \begin{cases} \ell'' = a_1'' c + \ell_1 \\ \ell'' = a_2'' e + \ell_2. \end{cases}$$

Mutta tällöin sekä c että e olisivat $\ell' - \ell''$:n tekijöitä eikä n olisikaan pyj(c,e). Näin ollen ℓ määräytyy yksikäsitteisesti luvuista ℓ_1 ja ℓ_2 ja nämä arvot voidaan taulukoida etukäteen. Kun on saatu ℓ sekä $\mathbf{s}_2^{(\ell_2)}(z)$, voidaan dekoodaus suorittaa.

Tässä menettelyssä tulee siirtoja enintään e + c – 2 kpl, mikä yleensä on huomattavasti vähemmän kuin ec – 1.

Kootaan yhteen yo. dekoodausalgoritmien antamat tulokset. Huomaa erityisesti tapaus t = 0, joka seuraa siitä, että $\mathbf{p}(z)$ ei ole $z^c - \overline{1}$:n tekijä (alaviite s. 55).

LAUSE 21. Fire-koodi, jonka generoijapolynomi on $\mathbf{g}(z) = (z^c - \overline{1})\mathbf{p}(z)$, yhtaikaisesti paljastaa purskevirheet, joiden pituus on $s \geq t$, ja korjaa purskevirheet, joiden pituus on $t \leq \deg(\mathbf{p}(z))$, mikäli $c \geq s + t - 1$. ■

Ns. *yleistetyn Fire-koodin* generoijapolynomi on muotoa

$$\mathbf{g}(z) = (z^c - \overline{1})\mathbf{p}_1(z)\cdots\mathbf{p}_h(z),$$

missä $\mathbf{p}_1(z),\ldots,\mathbf{p}_h(z)$ ovat erilaisia jaottomia (pää)polynomeja ja niiden jaksot $e_1,\ldots,e_h$ eivät ole c:n tekijöitä. Koodin pituus on lukujen $c,e_1,\ldots,e_h$ pienin yhteinen jaettava $n = \text{pyj}(c,e_1,\ldots,e_h)$. Tällaiselle koodille saadaan hyvin samantapaiset, mutta vielä nopeammat koodausmenettelyt kuin edellä "tavalliselle" Fire-koodille, ks. RAO & FUJIWARA. Viitteessä BLAHUT on hyvä esitys Fire-koodeista.

V LUKU
KOODIRAJAT

1. Yleistä

Koodin “hyvyyttä” sekä suorituskykyä voidaan mitata useillakin tunnusluvuilla. Jotta päästäisiin vertailemaan koodeja, skaalataan nämä luvut sopivasti. Näin saadaan (n,k)-koodin C *koodaussuhde*

$$\rho = \frac{k}{n}$$

sekä *virheenkorjaussuhde*

$$\lambda = \frac{d_{min}(C)}{n} .$$

ESIMERKKI. Toistokoodille (ks. Esimerkki s. 1) $\rho = 1/r$ ja $\lambda = 1/k$. Hamming-koodille (ks. **I.6**)

$$\rho = 1 - \frac{r(q-1)}{q^r - 1} \quad \text{ja} \quad \lambda = \frac{3(q-1)}{q^r - 1} .$$

Koodirajat ovat epäyhtälöitä, joilla arvioidaan yo. tunnuslukuja. Arviot ovat yleisiä. Asymptoottisissa rajoissa oletetaan koodin pituuden olevan suuri, jolloin mukaan tulee pieni virhetermi ε_n, joka lähenee nollaa pituuden n kasvaessa. Jatkossa (n,K,d)-koodilla tarkoitetaan n-pituista koodia, jonka koodisanojen lukumäärä on K ja minimietäisyys d. Koodisymbolien lukumäärää merkitään q:lla. Lineaariselle koodille on siis $K = q^k$, missä k on viestin pituus. Yleisesti merkitään $k = \log_q K$ (K:n q-kantainen logaritmi, ei välttämättä kokonaisluku) ja vastaavasti

$$\rho = \frac{\log_q K}{n} .$$

Yksinkertaisin koodiraja on

LAUSE 22. (SINGLETON-RAJA) (n,k,d)-koodille $d \le n - k + 1$ eli

$$\lambda \le 1 - \rho + \frac{1}{n}.$$

Asymptoottisesti $\lambda \le 1 - \rho + \varepsilon_n$.

Todistus. Asia on aivan ilmeinen sykliselle koodille, koska sen generoijapolynomi on koodipolynomi ja sen aste on $n - k$. Edelleen asia on ilmeinen myös lineaariselle koodille, sillä mitkä tahansa sen tarkistusmatriisin $n - k + 1$ saraketta ovat lineaarisesti riippuvat. Mutta Singleton-raja pätee aivan yleisesti. Merkitään K:lla kaikkien koodisanojen lukumäärää. Jos nyt olisi $K > q^{n-d+1}$, niin koodisanojen joukossa olisi kaksi eri sanaa, joiden $n - d + 1$ ensimmäistä symbolia ovat samat ja joiden etäisyys olisi näin enintään $d - 1$. Koska tämä ei ole mahdollista, on $K \le q^{n-d+1}$ eli $k \le n - d + 1$. ■

Koodeja, jotka ovat Singleton-rajalla, ts. $d = n - k + 1$, kutsutaan *MDS-koodeiksi**. Reed–Solomon-koodit (ks. **III.4**) ovat MDS-koodeja.

2. Ylärajoja

Singleton-rajan tapaisia ylärajoja tunnetaan useita. Johdetaan joitakin yksinkertaisia ylärajoja ja annetaan valmiina muutama lisää.

Plotkin-raja

Koodin $C = \{\mathbf{c}_1,\ldots,\mathbf{c}_K\}$ ns. *kokonaisetäisyys* on

$$d_{total}(C) = \sum_{i \ne j} d(\mathbf{c}_i, \mathbf{c}_j).$$

Jos q koodisymbolia ovat $a_1,\ldots,a_q$, niin merkitään K_{ij}:llä niiden koodisanojen lukumäärää, joiden i:s symboli on a_j. Silloin

$$K = \sum_{j=1}^{q} K_{ij} \quad \text{ja} \quad d_{total}(C) = \sum_{i=1}^{n}\sum_{j=1}^{q} K_{ij}(K - K_{ij}) = nK^2 - \sum_{i=1}^{n}\sum_{j=1}^{q} K_{ij}^2,$$

eikö totta. Toisaalta Cauchy–Schwarzin epäyhtälön (ks. peruskurssit) nojalla

---

* "MDS" = "maximum distance separable"

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{q} 1 \cdot K_{ij} \right)^2 \leq \left(\sum_{i=1}^{n} \sum_{j=1}^{q} 1^2 \right) \left(\sum_{i=1}^{n} \sum_{j=1}^{q} K_{ij}^2 \right)$$

eli

$$n^2 K^2 \leq nq \sum_{i=1}^{n} \sum_{j=1}^{q} K_{ij}^2,$$

(missä yhtäläisyysmerkki on voimassa tarkalleen silloin, kun kaikki K_{ij}:t saavat saman arvon (=K/q)). Näin ollen

$$d_{total}(C) \leq nK^2 \frac{q-1}{q}.$$

Koska ilmeisesti on

$$d_{min}(C) \leq \frac{d_{total}(C)}{K(K-1)},$$

saadaan

LAUSE 23. (PLOTKIN-RAJA) (n,K,d)-koodille

$$d \leq \frac{nK(q-1)}{(K-1)q} \qquad \text{eli} \qquad \lambda \leq \frac{K(q-1)}{(K-1)q} = \frac{q-1}{q(1-q^{-n\rho})}. \quad \blacksquare$$

Näin ollen, jos halutaan saavuttaa tietty virheenkorjaussuhde $\lambda > 1 - 1/q$, pitää olla

$$\rho \leq \frac{1}{n} \log_q \frac{q\lambda}{q\lambda - q + 1}.$$

Tietyssä mielessä asymptoottinen Plotkin-raja on siis $\lambda \leq 1 - 1/q + \varepsilon_n$. Toisaalta Plotkin-rajaa ovelasti käyttäen saadaan parempiakin rajoja. Ilmeisesti C :ssä on ainakin $K/q^T = q^{k-T}$ sellaista koodisanaa, joiden T ensimmäistä symbolia ovat samat. Valitaan nimenomaan $K' = \lceil q^{k-T} \rceil$ sellaista sanaa ja muodostetaan niistä n – T-pituinen koodi C' poistamalla kustakin sanasta T ensimmäistä symbolia. C' on ns. C :stä *lyhennetty koodi.* Tällöin $d_{min}(C) \leq d_{min}(C')$. Valitsemalla nyt T sopivasti saadaan todistettua

LAUSE 24. (PARANNETTU PLOTKIN-RAJA) (n,q^k,d)-koodille C

$$d \le \left(1 - \frac{1}{q}\right)(n - k) + \left(1 - \frac{1}{q}\right)\log_q k + \frac{n - k + \log_q k}{k - q}(q - 1).$$

Asymptoottisesti $\boxed{\lambda \le \left(1 - \frac{1}{q}\right)(1 - \rho) + \varepsilon_n.}$

Todistus. Merkitään $\ell = \log_q k$ ja valitaan koodia C lyhennettäessä T:ksi $\lceil k - \ell \rceil$. Sovelletaan näin saadulle (n – T,K′,d′)-koodille C' Plotkin-rajaa ja arvioidaan ylöspäin:

$$d_{min}(C) \le d_{min}(C') \le \frac{(n - T)K'(q - 1)}{(K' - 1)q}$$

$$= \left(1 - \frac{1}{q}\right)(n - T) + \frac{(n - T)(q - 1)}{(K' - 1)q}$$

$$\le \left(1 - \frac{1}{q}\right)(n - k) + \left(1 - \frac{1}{q}\right)\log_q k + \frac{n - k + \log_q k}{k - q}(q - 1).$$

Asymptoottinen raja seuraa tästä. ■

Hamming-raja

R-säteisessä koodisanakeskisessä pallossa B(**c**,R) on sanoja

$$V(R) = \sum_{i=0}^{R} \binom{n}{i} (q - 1)^i$$

kpl, eikö totta. Tätä yksinkertaista tietoa käyttäen saadaan

LAUSE 25. (HAMMING-RAJA) Jos (n,K,d)-koodi C korjaa t virhettä, niin

$$\rho \le 1 - \frac{1}{n}\log_q V(t).$$

Todistus. Jos nyt olisi $KV(t) > q^n$, niin $d < 2t + 1$ eikä C korjaisi t virhettä. Siispä $KV(t) \le q^n$ eli $V(t) \le q^{n-k}$ eli $\log_q V(t) \le n - k$. ■

Hamming-koodit ovat Hamming-rajalla. Hamming-rajan asymptoottisen muodon saamiseksi pitää arvioida V(t):tä. Merkitään lyhyyden vuoksi

$$A_i = \binom{n}{i}(q-1)^i \qquad (i = 0,1,\ldots,t).$$

Silloin $V(t) \geq A_t$. Ns. heikon Stirlingin kaavan (ks. kurssi 73265 Symbolinen analyysi 1) mukaan

$$\frac{j^j}{e^{j-1}} \leq j! \leq \frac{j^{j+1}}{e^{j-1}}.$$

Sovelletaan tätä A_t:n binomikertoimeen:

$$A_t \geq \frac{1}{et(n-t)}\frac{(n/t-1)^t}{(1-t/n)^n}(q-1)^t = \frac{1}{et(n-t)}\frac{1}{(1-t/n)^{n-t}}\left(\frac{q-1}{t/n}\right)^t.$$

Silloin nähdään, että

$$\frac{1}{n}\log_q A_t \geq \frac{1}{n}\log_q \frac{1}{et(n-t)} + H_q\left(\frac{t}{n}\right),$$

missä $H_q(x)$ on entropiafunktio

$$H_q(x) = -(1-x)\log_q(1-x) - x\log_q\frac{x}{q-1}$$

(vrt. **VII.3**). Tässä $-\frac{1}{n}\log_q\left(et(n-t)\right) \to 0$, kun $n \to \infty$, joten se "kuuluu osaan ε_n". Nyt $\frac{\lambda}{2} - \frac{3}{2n} < \frac{t}{n} < \frac{\lambda}{2}$ ja entropiafunktio on jatkuva funktio, joten Hamming-rajan asymptoottinen muoto on

$$\boxed{\rho \leq 1 - H_q(\lambda/2) + \varepsilon_n.}$$

Muita ylärajoja

Yhdistämällä Plotkin-rajaa sekä Hamming-rajaa johdettaessa käytettyjä menetelmiä saadaan vahvempi raja, ns. *Elias-raja* (asymptoottinen)

$$\boxed{\rho \leq 1 - H_q\left(\left(1-\frac{1}{q}\right)\left(1-\sqrt{1-\lambda\frac{q}{q-1}}\right)\right) + \varepsilon_n}$$

(ks. esimerkiksi VAN LINT). Edelleen menetelmiä terästämällä saadaan eräs parhaita tunnetuista asymptoottisista ylärajoista, ns. *McEliece–Rodemich–Rumsey–Welch-raja*

$$\boxed{\rho \leq H_q\left(1 - \frac{1}{q} - \lambda\left(1-\frac{2}{q}\right) - \frac{2}{q}\sqrt{\lambda(1-\lambda)(q-1)}\right) + \varepsilon_n}$$

(binäärisen koodin tapauksessa ks. esimerkiksi VAN LINT, yleisen tapauksen todistikin* muuten suomalaismatemaatikko Matti Aaltonen).

3. Alarajoja

Ylärajat määrittävät alueita, joilla mielivaltaisen (n,K,d)-koodin parametrikolmikko (n,K,d) (tai vastaavasti (n,ρ,λ)) ei voi olla. Alarajojen tarkoituksena on antaa alueita, joilla (n,K,d)-koodeja on.

Gilbert-raja

LAUSE 26. (GILBERT-RAJA) Jos

$$V(d-1) < \frac{q^n}{K-1},$$

niin on olemassa (n,K,d)-koodi.

Todistus. Valitaan kaksi n-pituista sanaa $\mathbf{c}_1$ ja $\mathbf{c}_2$, joille $d(\mathbf{c}_1,\mathbf{c}_2) = d$, ja merkitään $C_2 = \{\mathbf{c}_1,\mathbf{c}_2\}$. Jatketaan tästä rekursiivisesti lisäämällä koodisanoja yksi kerrallaan. Jos käsillä on koodi $C_{i-1} = \{\mathbf{c}_1,\ldots,\mathbf{c}_{i-1}\}$ ja on sellainen sana $\mathbf{c}_i$, joka ei ole joukossa $\bigcup_{j=1}^{i-1} B(\mathbf{c}_j, d-1)$, niin otetaan sana mukaan koodiin, ts. valitaan seuraavaksi koodiksi $C_i = \{\mathbf{c}_1,\ldots,\mathbf{c}_i\}$. Tämä on mahdollista niin kauan kuin $i \le K$, sillä mainitun joukon komplementissa on tällöin ainakin $q^n - (i-1)V(d-1) \ge 1$ sanaa. Lopulta saadun koodin minimietäisyys on ilmeisestikin d. ■

Asymptoottisen rajan saamiseksi palataan Hamming-rajan johdossa tarvittuihin käsitteisiin. Ensinnäkin todetaan, että

$$A_{i+1} = \frac{(n-i)(q-1)}{i+1} A_i \qquad (i = 0,\ldots,d-1).$$

Mutta jos $\lambda \le 1 - 1/q$ eli $dq \le n(q-1)$, kuten oletetaan, niin

$$\frac{(n-i)(q-1)}{i+1} = \frac{i + n(q-1) - iq}{i+1} \ge \frac{i + n(q-1) - (d-1)q}{i+1} \ge 1.$$

Jono $A_1,A_2,\ldots,A_d$ on näin ollen kasvava ja $V(d) \le (d+1)A_d$ eli

$$\log_q V(d) \le \log_q (d+1) + \log_q A_d,$$

---

* AALTONEN, M.J.: Linear Programming Bounds for Tree Codes. *IEEE Transactions on Information Theory* **25** (1979), 85–90

mikäli $\lambda \le 1 - 1/q$. Selvästi $\frac{1}{n}\log_q(d+1) \to 0$, kun $n \to \infty$, joten se "kuuluu osaan ε_n". Sovelletaan heikkoa Stirlingin kaavaa A_d:n binomikertoimeen:

$$A_d \le \frac{n}{e}\frac{(n/d-1)^d}{(1-d/n)^n}(q-1)^d = \frac{n}{e}\frac{1}{(1-d/n)^{n-d}}\left(\frac{q-1}{d/n}\right)^d.$$

Silloin nähdään, että

$$\frac{1}{n}\log_q A_d \le \frac{1}{n}\log_q \frac{n}{e} + H_q\left(\frac{d}{n}\right).$$

Selvästi jälleen $\frac{1}{n}\log_q \frac{n}{e} \to 0$, kun $n \to \infty$, joten sekin "kuuluu osaan ε_n".

Muodostetaan sitten Gilbert-rajan todistuksessa esitetyllä menetelmällä mahdollisimman suuri koodi. Silloin saadulle (n,K,d)-koodille

$$V(d-1) \ge \frac{q^n}{K} = q^{n-k},$$

koska muuten konstruktiota voitaisiin jatkaa, ja näin

$$1 - \rho \le \frac{1}{n}\log_q V(d-1) \le \frac{1}{n}\log_q V(d).$$

Asymptoottisesti siis jokaiselle kyllin suurelle n:n arvolle on olemassa n-pituinen koodi, jolle $\boxed{\rho \ge 1 - H_q(\lambda) + \varepsilon_n.}$

Konstruktio voidaan tehdä myös siten, että saatu koodi on lineaarinen. Asymptoottinen muoto on luonnollisesti tällöin sama kuin yllä.

LAUSE 27. (GILBERT-RAJA LINEAARISILLE KOODEILLE) Jos

$$V(d-1) < q^{n-k+1},$$

niin on olemassa lineaarinen (n,k,d)-koodi.

Todistus. Valitaan n-pituinen sana $\mathbf{c}_1$, jolle $w(\mathbf{c}_1) = d$, ja merkitään $C_1 = \{ f\mathbf{c}_1 \mid f \in \mathbb{F} \}$. Jatketaan tästä rekursiivisesti lisäämällä koodisanoja. Jos käsillä on koodi C_{i-1} ja on sellainen sana $\mathbf{c}_i$, joka ei esiinny joukossa $\bigcup_{\mathbf{c}\in C_{i-1}} B(\mathbf{c}, d-1)$, niin otetaan koodiin mukaan kaikki $\mathbf{c}_i$:n generoimat sanat, ts. valitaan seuraavaksi koodiksi

$$C_i = \{ f\mathbf{c}_i + \mathbf{c} \mid f \in \mathbb{F} \text{ ja } \mathbf{c} \in C_{i-1} \}.$$

Tämä on mahdollista niin kauan kun $i \le k$, sillä mainitun joukon komplementissa on tällöin ainakin $q^n - q^{i-1}V(d-1) \ge 1$ sanaa. Lopulta saadun

koodin minimietäisyys on ilmeisestikin d. ■

Varshamov-raja

Lineaarisille koodeille on parempikin alaraja:

LAUSE 28. (VARSHAMOV-RAJA) Jos

$$\sum_{j=0}^{d-2}\binom{n-1}{j}(q-1)^j < q^{n-k},$$

niin on olemassa lineaarinen (n,k)-koodi, jonka minimietäisyys on vähintään d.

Todistus. Valitaan n – k kpl lineaarisesti riippumattomia n – k-vektoreita ja muodostetaan niitä sarakkeina käyttäen neliömatriisi $\mathbf{H}_{n-k}$. Jatketaan tästä rekursiivisesti lisäämällä matriisiin yksi kerrallaan sarakkeita. Tarkoitus on, että aina mitkä tahansa matriisin d – 1 saraketta ovat lineaarisesti riippumattomia. Jos käsillä olevassa matriisissa $\mathbf{H}_{i-1}$ on i – 1 saraketta, niin niistä voidaan muodostaa enintään d – 2 sarakkeen muodostamia lineaariyhdelmiä korkeintaan $\sum_{j=0}^{d-2}\binom{i-1}{j}(q-1)^j$ kpl. Konstruktiota voidaan näin ollen jatkaa niin kauan kuin $i \le n$, sillä tällöin

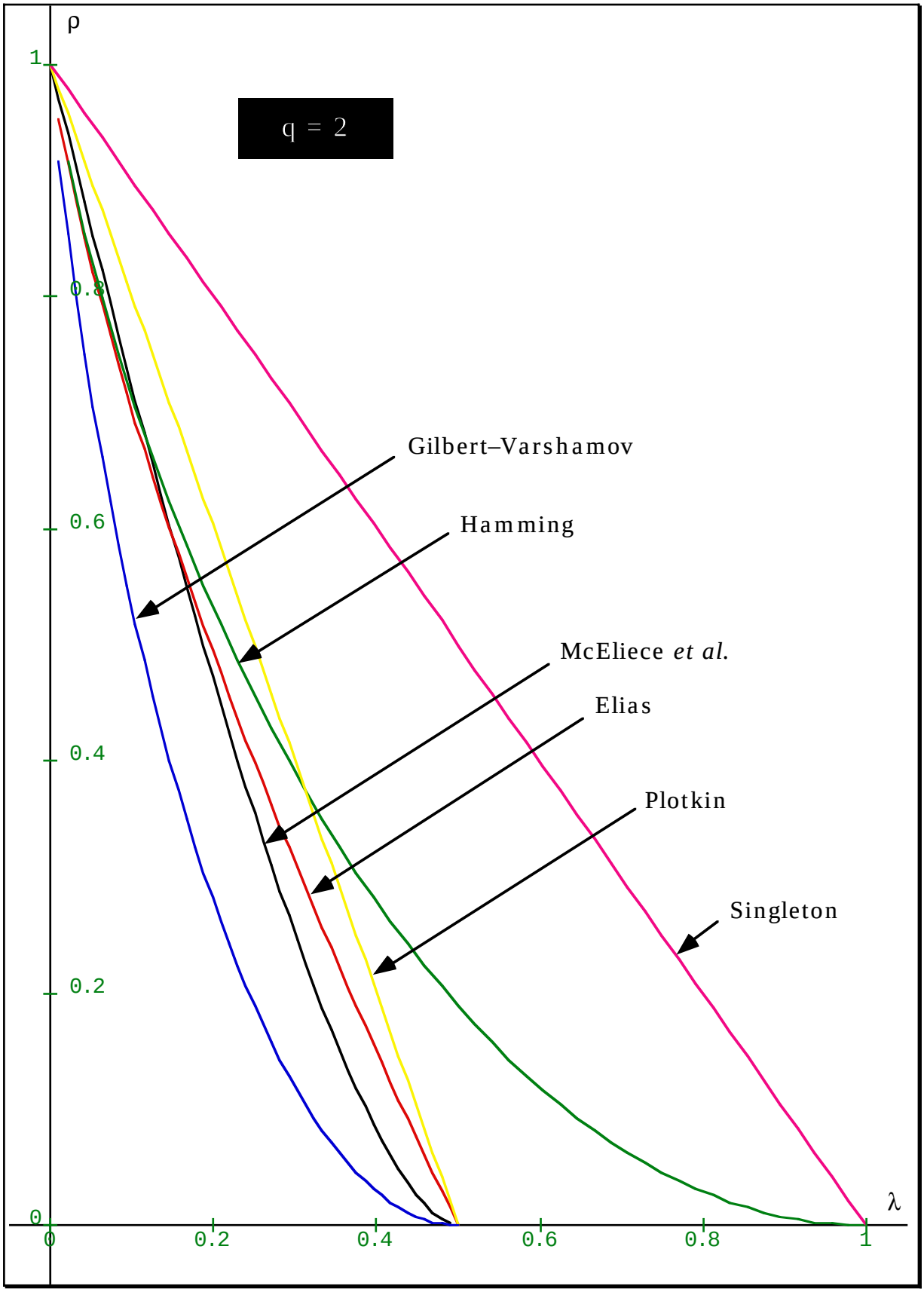
$$\sum_{j=0}^{d-2}\binom{i-1}{j}(q-1)^j \le \sum_{j=0}^{d-2}\binom{n-1}{j}(q-1)^j < q^{n-k}.$$

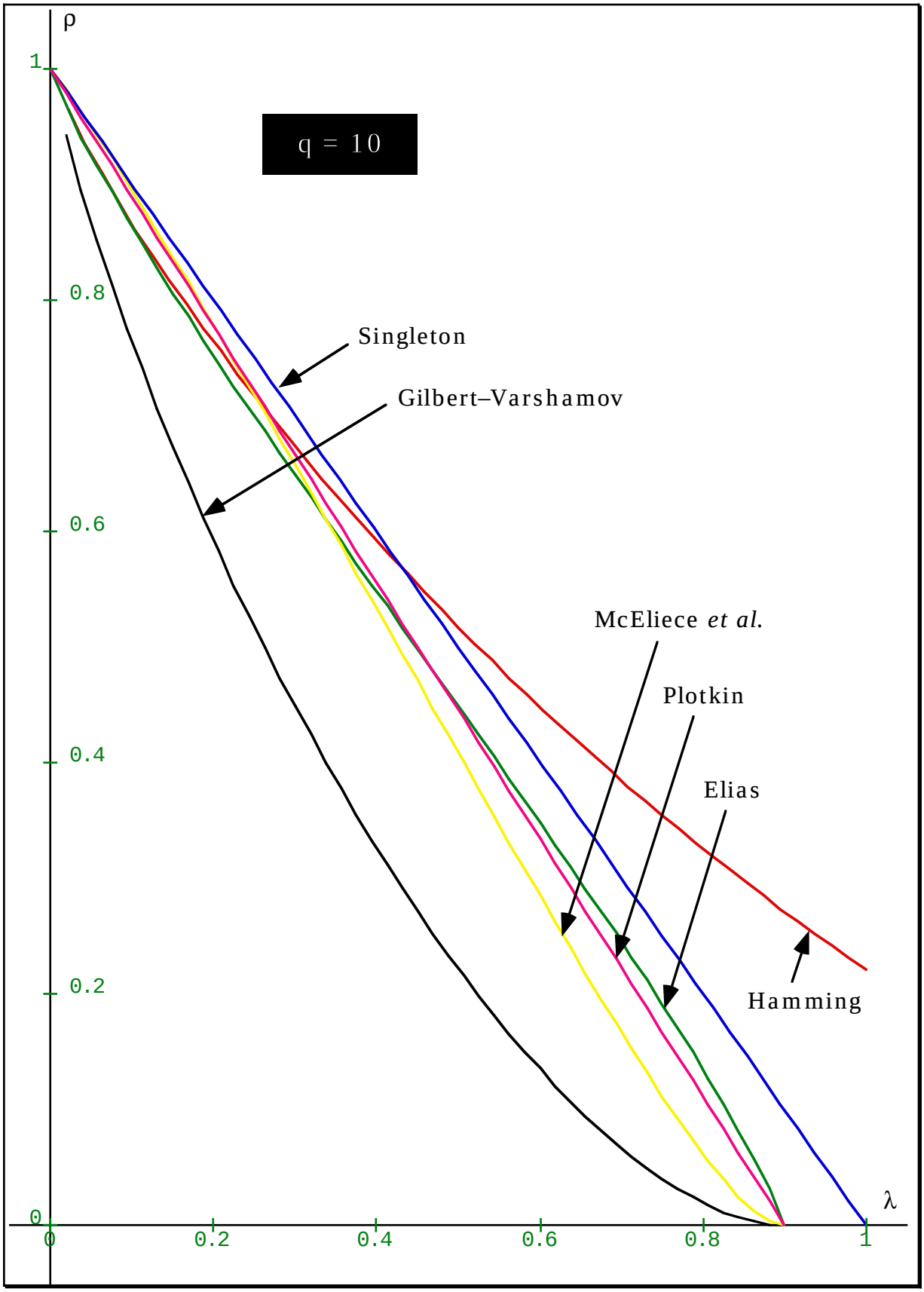
Lauseen **2** nojalla saatua tarkistusmatriisia vastaavan lineaarisen (n,k)-koodin minimietäisyys on ainakin d. ■

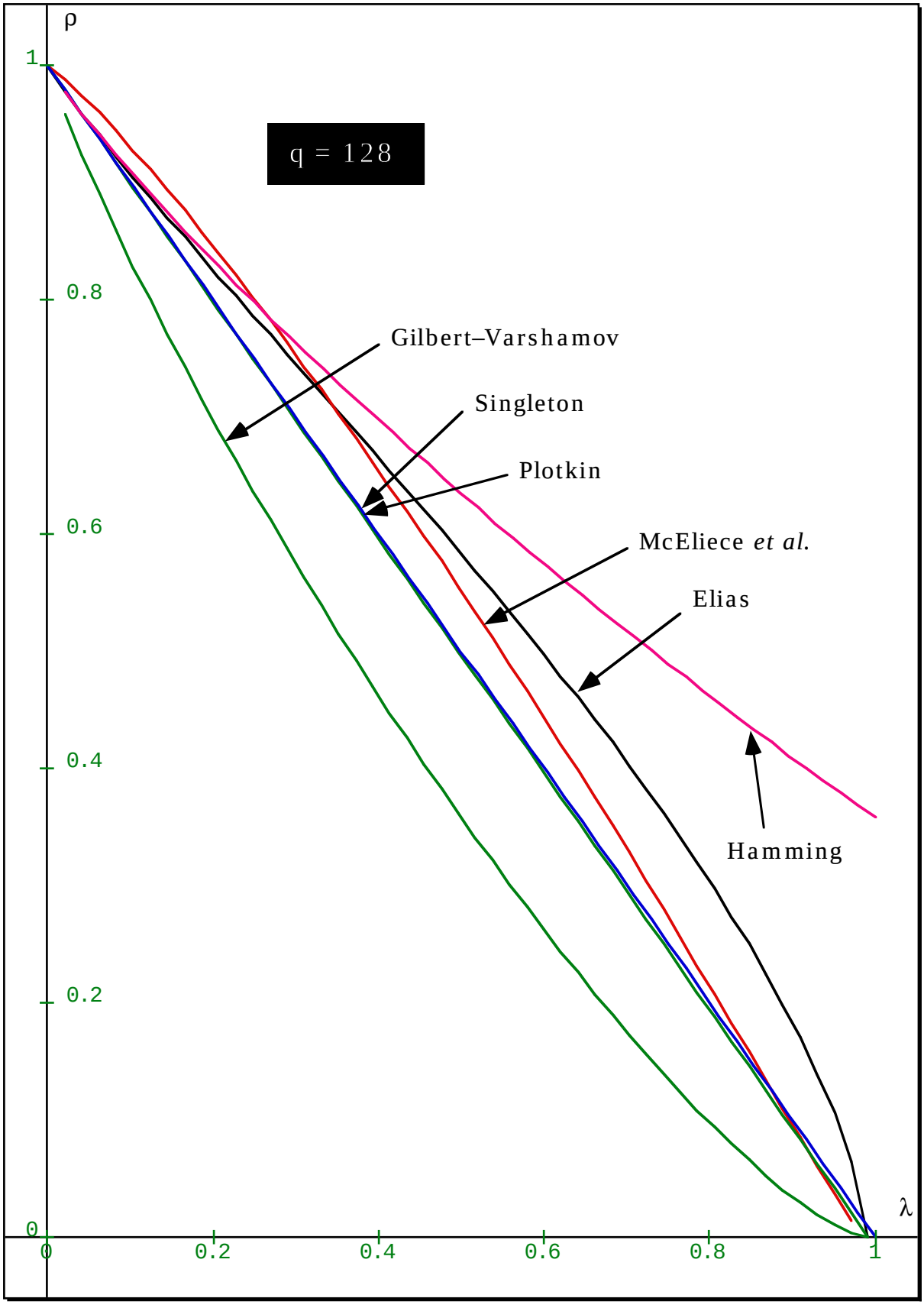
Koska $V(d-1) = q\sum_{j=0}^{d-2}\binom{n-1}{j}(q-1)^j + \binom{n-1}{d-1}(q-1)^{d-1}$ (totea!), Varshamov-raja on todellakin vahvempi kuin Gilbert-raja (eli Lause **27** seuraa Lauseesta **28**). Varshamov-rajan asymptoottinen muoto on sama kuin Gilbert-rajan (josta syystä mainittu raja tunnetaan *Gilbert–Varshamov-rajana*). Tämä raja on mahdollista saavuttaa yhtenäisellä koodiperheellä (esimerkiksi ns. Goppa-koodeilla, ks. MCELIECE).

4. Rajojen vertailua

Rajojen keskinäinen paikka ilmenee parhaiten kuviosta, johon ne kaikki on piirretty. Otetaan mukaan vain asymptoottiset rajat ja “unohdetaan” termi ε_n. Rajat on piirrettu Maplella.


ρ
1
0.8
0.6
0.4
0.2
0
q = 2
Gilbert–Varshamov
Hamming
McEliece *et al.*
Elias
Plotkin
Singleton
λ
0
0.2
0.4
0.6
0.8
1


ρ
1
0.8
0.6
0.4
0.2
0
q = 10
Singleton
Gilbert–Varshamov
McEliece *et al.*
Plotkin
Elias
Hamming
λ
0
0.2
0.4
0.6
0.8
1


ρ
1
0.8
0.6
0.4
0.2
0
q = 128
Gilbert–Varshamov
Singleton
Plotkin
McEliece *et al.*
Elias
Hamming
λ
0
0.2
0.4
0.6
0.8
1

"It is interesting to observe that all the codes in this chapter somehow relate to convolution. Some wag has said that communications engineers don't know how to do much else. He may be right."

ANDERSON & MOHAN (Luvun 4 esipuhe)

VI LUKU

KONVOLUUTIOKOODAUS

1. Ekskursio: Lineaariset järjestelmät

Lineaarinen järjestelmä S muuntaa äärettömän $\mathbb{F}^k$:n vektorien syötejonon

$$[\ldots,\mathbf{a}_{-2},\mathbf{a}_{-1},\mathbf{a}_0,\mathbf{a}_1,\mathbf{a}_2,\ldots]$$

äärettömäksi $\mathbb{F}^n$:n vektorien vastejonoksi

$$[\ldots,\mathbf{b}_{-2},\mathbf{b}_{-1},\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2,\ldots]$$

konvoloiden sen k×n-matriisien jonon kanssa:

$$[\ldots,\mathbf{a}_{-2},\mathbf{a}_{-1},\mathbf{a}_0,\mathbf{a}_1,\mathbf{a}_2,\ldots] \longrightarrow \boxed{S} \longrightarrow [\ldots,\mathbf{b}_{-2},\mathbf{b}_{-1},\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2,\ldots]$$

missä

$$\mathbf{b}_i = \sum_{j=-\infty}^{\infty} \mathbf{a}_j \mathbf{H}_{i-j}(i) = \sum_{\ell=-\infty}^{\infty} \mathbf{a}_{i-\ell} \mathbf{H}_\ell(i) \quad \text{(konvoluutio).}$$

Jotta summa olisi määritelty pitää lisäksi huolehtia siitä, että sen termeistä vain äärellisen monta on nollavektorista eroavaa. (Nimitys "lineaarinen" tulee siitä, että syötejonojen summan vastejono on yhteenlaskettavien vasteiden summa ja skalaarilla kerrotun syötejonon vaste on alkuperäisen syötteen vaste kerrottuna samalla skalaarilla. Jokainen nämä ehdot toteuttava järjestelmä voidaan toisaalta pukea yo. muotoon.)

i:stä riippuva matriisijono

$$[\ldots,\mathbf{H}_{-2}(i),\mathbf{H}_{-1}(i),\mathbf{H}_0(i),\mathbf{H}_1(i),\mathbf{H}_2(i),\ldots]$$

on ns. *impulssivaste*. Nimitys tulee siitä, että jos syötejono on impulssi

$$[\ldots,\mathbf{0},\mathbf{0},\underset{\uparrow}{\mathbf{e}_u},\mathbf{0},\mathbf{0},\ldots],$$
$$= \mathbf{a}_v$$

missä $\mathbf{e}_u$ on vektori, jonka u:s komponentti on $\overline{1}$ ja muut nolla-alkioita, niin tulostejonossa on

$$\mathbf{b}_i = \mathbf{e}_u\mathbf{H}_{i-v}(i) = \mathbf{H}_{i-v}(i)\text{:n u:s rivi.}$$

Järjestelmän ns. *matriisiesitys* saadaan ottamalla $\mathbf{H}_j(i)$:t äärettömän matriisin lohkoiksi ja samoin $\mathbf{a}_j$:t sekä $\mathbf{b}_j$:t äärettömien vektorien lohkoiksi:

$$(\ldots,\mathbf{a}_{-1},\mathbf{a}_0,\mathbf{a}_1,\ldots)\left(\begin{array}{c|c|c|c|c} \ddots & \vdots & \vdots & \vdots & \iddots \\ \hline \cdots & \mathbf{H}_0(-1) & \mathbf{H}_1(0) & \mathbf{H}_2(1) & \cdots \\ \hline \cdots & \mathbf{H}_{-1}(-1) & \mathbf{H}_0(0) & \mathbf{H}_1(1) & \cdots \\ \hline \cdots & \mathbf{H}_{-2}(-1) & \mathbf{H}_{-1}(0) & \mathbf{H}_0(1) & \cdots \\ \hline \iddots & \vdots & \vdots & \vdots & \ddots \end{array}\right) = (\ldots,\mathbf{b}_{-1},\mathbf{b}_0,\mathbf{b}_1,\ldots).$$

Lineaarinen järjestelmä S on *stationäärinen*, jos matriisit

$$\ldots,\mathbf{H}_{-2}(i),\mathbf{H}_{-1}(i),\mathbf{H}_0(i),\mathbf{H}_1(i),\mathbf{H}_2(i),\ldots$$

eivät riipu "ajasta" i, merkitään $\mathbf{H}_j(i) = \mathbf{H}_j$. Edelleen S on *kausaalinen*, jos

$$\cdots = \mathbf{H}_{-2}(i) = \mathbf{H}_{-1}(i) = \mathbf{O}_{k\times n},$$

ts. vaste ei riipu syötteen tulevista arvoista. Jatkossa tarkastellaan vain stationäärisiä ja kausaalisia järjestelmiä S. Muita ei juurikaan käytetä konvoluutiokoodauksessa. Tällaisen järjestelmän matriisiesitys on muotoa

$$(\ldots,\mathbf{a}_{-1},\mathbf{a}_0,\mathbf{a}_1,\ldots)\left(\begin{array}{c|c|c|c|c} \ddots & \vdots & \vdots & \vdots & \iddots \\ \hline \cdots & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \cdots \\ \hline \cdots & \mathbf{O} & \mathbf{H}_0 & \mathbf{H}_1 & \cdots \\ \hline \cdots & \mathbf{O} & \mathbf{O} & \mathbf{H}_0 & \cdots \\ \hline \iddots & \vdots & \vdots & \vdots & \ddots \end{array}\right) = (\ldots,\mathbf{b}_{-1},\mathbf{b}_0,\mathbf{b}_1,\ldots)$$

ja konvoluutio on

$$\mathbf{b}_i = \sum_{j=-\infty}^{i}\mathbf{a}_j\mathbf{H}_{i-j} = \sum_{\ell=0}^{\infty}\mathbf{a}_{i-\ell}\mathbf{H}_\ell.$$

Jotta konvoluutiosumma olisi määritelty, sovitaan että syötejonot ovat muotoa

$$[\ldots,\mathbf{0},\mathbf{0},\mathbf{a}_0,\mathbf{a}_1,\mathbf{a}_2,\ldots],$$

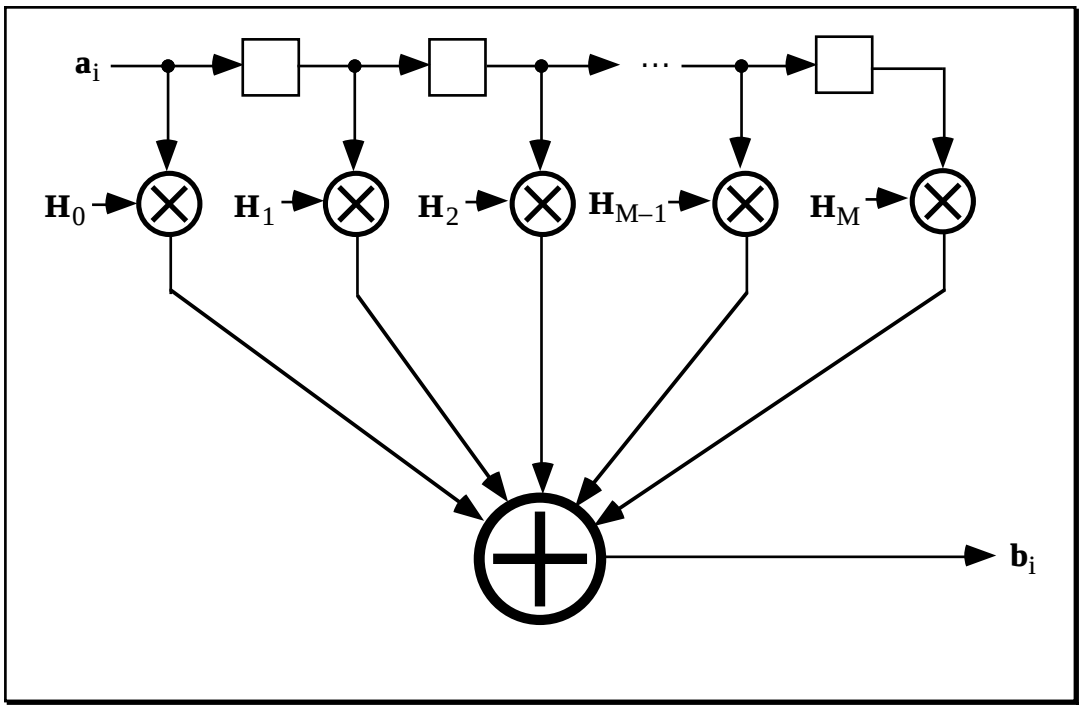
ts. varsinaiset syötteet alkavat vasta "ajanhetkellä" i = 0. Silloin konvoluutio on muotoa

$$\mathbf{b}_i = \sum_{j=0}^{i} \mathbf{a}_j \mathbf{H}_{i-j} = \sum_{\ell=0}^{i} \mathbf{a}_{i-\ell} \mathbf{H}_\ell \quad (i = 0,1,2,\ldots).$$

Jos jollekin arvolle M on $\mathbf{H}_{M+1} = \mathbf{H}_{M+2} = \cdots = \mathbf{O}_{k\times n}$, sanotaan lineaarista järjestelmää *äärelliseksi* eli *polynomiaaliseksi.* Jos taas jono $\mathbf{H}_0,\mathbf{H}_1,\mathbf{H}_2,\ldots$ on jostain rajasta eteenpäin jaksollinen, kutsutaan järjestelmää *periodiseksi* eli *rationaaliseksi.* (Huomaa, että polynomiaalinen järjestelmä on myös rationaalinen.) Konvoluutiokooderit ovat enimmäkseen polynomiaalisia. Takaisinkytkennällä saadaan aikaan myös rationaalisia koodereita. Polynomiaalisen lineaarisen järjestelmän matriisiesityksen matriisi on lohkonauhamuotoa:

$$\begin{pmatrix}
\ddots & \vdots & \vdots & \vdots & \vdots & \iddots \\
\cdots & \mathbf{H}_{M-1} & \mathbf{H}_M & \mathbf{O} & \mathbf{O} & \cdots \\
\cdots & \mathbf{H}_{M-2} & \mathbf{H}_{M-1} & \mathbf{H}_M & \mathbf{O} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & \mathbf{O} & \mathbf{H}_0 & \mathbf{H}_1 & \mathbf{H}_2 & \cdots \\
\cdots & \mathbf{O} & \mathbf{O} & \mathbf{H}_0 & \mathbf{H}_1 & \cdots \\
\cdots & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{H}_0 & \cdots \\
\iddots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{pmatrix}.$$

Polynomiaalinen järjestelmä voidaan esittää myös *siirtorekisterimuodossa*



missä □ on viive, ⊕ on yhteenlasku ja ⊗ on kertolasku. Vektoraaliset rekisterit esitetään usein komponentteihinsa hajoitettuina.

Matriisiesitys on ns. aikatason esitys. Taajuustasoon päästään ottamalla z-muunnokset syötejonosta, vastejonosta sekä impulssivasteesta:

$$\mathbf{a}(z) = \sum_{j=0}^{\infty} \mathbf{a}_j z^j, \quad \mathbf{b}(z) = \sum_{j=0}^{\infty} \mathbf{b}_j z^j \quad \text{ja} \quad \mathbf{H}(z) = \sum_{j=0}^{\infty} \mathbf{H}_j z^j.$$

H(z) on ns. *siirtomatriisi.* Huomaa, että jos järjestelmä on polynomiaalinen, niin **H**(z) on matriisikertoiminen polynomi eli polynomialkioinen matriisi (tästä nimi ”polynomiaalinen”). Jos taas järjestelmä on rationaalinen ja jakson pituus on L, niin **H**(z) on muotoa

$$\mathbf{H}(z) = \mathbf{P}(z) + \mathbf{Q}(z) \sum_{j=0}^{\infty} \mathbf{I}_n z^{Lj},$$

missä **P**(z) ja **Q**(z) ovat matriisikertoimisia polynomeja. Toisaalta kertomalla havaitaan, että

$$(\overline{1} - z^L) \sum_{j=0}^{\infty} z^{Lj} = \overline{1} \quad \text{eli} \quad \sum_{j=0}^{\infty} z^{Lj} = \frac{\overline{1}}{\overline{1} - z^L}.$$

Näin ollen rationaalisen järjestelmän siirtomatriisin alkiot ovat $\mathbb{F}$[z]:n polynomien formaalisia osamääriä* eli rationaalifunktioita (tästä nimi “rationaalinen”).

Koska konvoluutio vastaa z-muunnosten puolella kertolaskua, on

$$\mathbf{b}(z) = \mathbf{a}(z)\mathbf{H}(z).$$

Lineaarisen järjestelmän sanotaan olevan *kääntyvä* eli *invertoituva,* jos sen siirtomatriisi **H**(z) on täysiranginen. Tällöin syöte määräytyy yksikäsitteisesti vasteesta. Kooderille kääntyvyys on luonnollinen vaatimus.

Vielä todetaan, että lineaarinen järjestelmä on *kytkemätön,* jos sen siirtomatriisin **H**(z) kussakin sarakkeessa on vain (enintään) yksi $\overline{0}$:sta eroava alkio (sama ominaisuus on silloin myös matriiseilla $\mathbf{H}_0, \mathbf{H}_1, \ldots$). Kytkemättömässä järjestelmässä syötteen komponenteilla eli “kanavilla” ei ole yhteisvaikutusta vasteeseen.

---

* Näillä osamäärillä lasketaan tavalliseen tapaan. Erityisesti on huomattava, että jos $\mathbf{p}(z), \mathbf{q}(z) \neq \overline{0}$ ja $\mathbf{r}(z) \neq \overline{0}$ ovat $\mathbb{F}$[z]:n polynomeja, niin formaaliset rationaalifunktiot $\frac{\mathbf{p}(z)}{\mathbf{q}(z)}$ ja $\frac{\mathbf{r}(z)\mathbf{p}(z)}{\mathbf{r}(z)\mathbf{q}(z)}$ samaistetaan täysin. Samoin rationaalifunktio $\frac{\mathbf{p}(z)}{\overline{1}}$ ja polynomi **p**(z) samaistetaan. Kunnan $\mathbb{F}$ kaikkien rationaalifunktioiden joukko on näin itse asiassa kunta, ns. $\mathbb{F}$:n rationaalifunktioiden kunta, merkitään $\mathbb{F}(z)$.

2. Konvoluutiokoodaus

Varsinaisen koodin sijasta tarkastellaan vain koodausta. *Konvoluutiokoodaus* on koodaus, joka suoritetaan polynomiaalisella tai rationaalisella kääntyvällä lineaarisella järjestelmällä, ns. *konvoluutiokooderilla*. Useimmiten konvoluutiokooderit ovat polynomiaalisia. Viestilohkot syötetään vektoreina kooderiin. Vastevektorit taas kiedotaan yhteen (ks. **IV.3**).

Konvoluutiokooderissa on muisti, joka aiheuttaa redundanssia paitsi "paikallisesti" eli koodisanan sisällä (kuten lohkokoodeissa) myös "ajallisesti" eli peräkkäisten (kiedottujen) koodisanojen kesken. Tämä pitää ottaa huomioon virheitä korjaavassa dekoodauksessa.

Polynomiaaliselle konvoluutiokooderille on $\mathbf{H}_{M+1} = \mathbf{H}_{M+2} = \cdots = \mathbf{O}_{k\times n}$. (M on ns. *muistin pituus*). Silloin konvoluutio on muotoa

$$\mathbf{b}_i = \sum_{\ell=0}^{M} \mathbf{a}_{i-\ell}\mathbf{H}_\ell \quad (i = 0,1,2,\ldots).$$

Vektorijono $\langle \mathbf{a}_{i-1},\ldots,\mathbf{a}_{i-M}\rangle$ muodostaa kooderin ns. *tilan* hetkellä i. Saatuaan uuden vektorin $\mathbf{a}_i$ syötteenä kooderin tila vaihtuu

$$\langle \mathbf{a}_{i-1},\ldots,\mathbf{a}_{i-M}\rangle \rightarrow \langle \mathbf{a}_i,\ldots,\mathbf{a}_{i-M+1}\rangle$$

ja vastevektori

$$\mathbf{b}_i = \sum_{\ell=0}^{M} \mathbf{a}_{i-\ell}\mathbf{H}_\ell$$

tulostuu. Koodaus voi olla myös *systemaattista,* valitaan vain matriisit $\mathbf{H}_0$, ...,$\mathbf{H}_M$ vaikkapa siten, että ne ovat muotoa

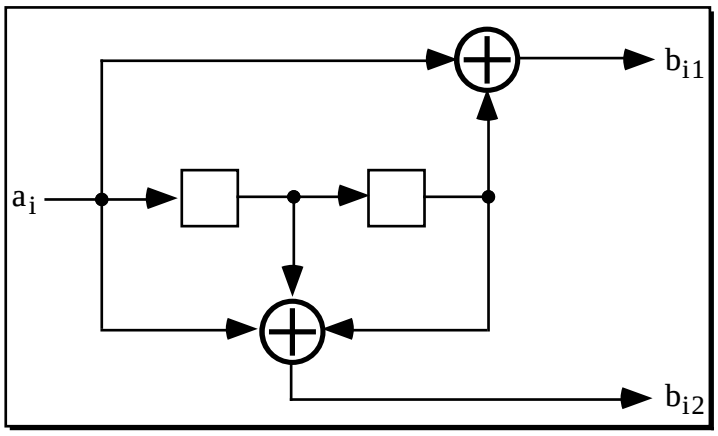
$$\mathbf{H}_0 = \left(\mathbf{I}_k \middle| \mathbf{P}_0\right), \mathbf{H}_1 = \left(\mathbf{O}_k \middle| \mathbf{P}_1\right), \ldots, \mathbf{H}_M = \left(\mathbf{O}_k \middle| \mathbf{P}_M\right),$$

jolloin $\mathbf{b}_i = (\mathbf{a}_i, \mathbf{c}_i)$.

HUOM! Myös rationaaliselle kooderille voidaan määritellä tilat, kun se toteutetaan takaisinkytkentää käyttäen. Jatkossa tarkastellaan kuitenkin vain polynomiaalisia koodereita.

Mahdollisia tiloja on äärellinen määrä, merkitään $\langle \mathbf{0},\ldots,\mathbf{0}\rangle = S_1,\ldots,S_N$. Tilasiirtymät voidaan esittää tällöin graafisesti muodossa ⑧ $\xrightarrow{\mathbf{a}/\mathbf{b}}$ ④ (siirtymä tilasta S_8 tilaan S_4 syötteellä **a** tulostaen **b**). Kun kaikki mahdolliset tilasiirtymät esitetään yhdessä, saadaan ns. *tiladiagrammi.*

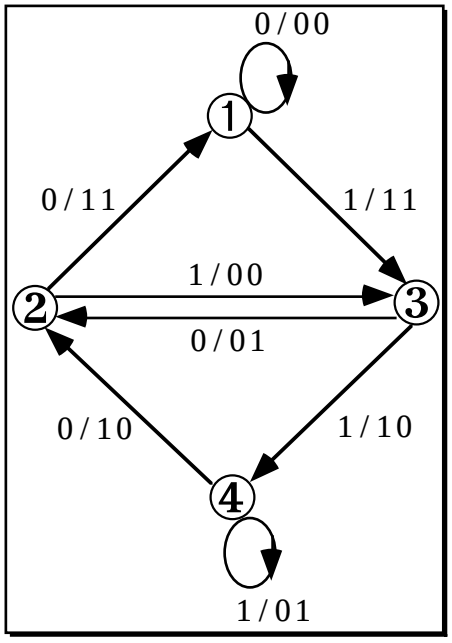
ESIMERKKI. Tarkastellaan binääristä polynomiaalista konvoluutiokooderia, jonka siirtorekisteri on



Tässä k = 1, n= 2 ja M = 2. Siirtomatriisi on

$$\mathbf{H}(z) = (1 + z^2, 1 + z + z^2).$$

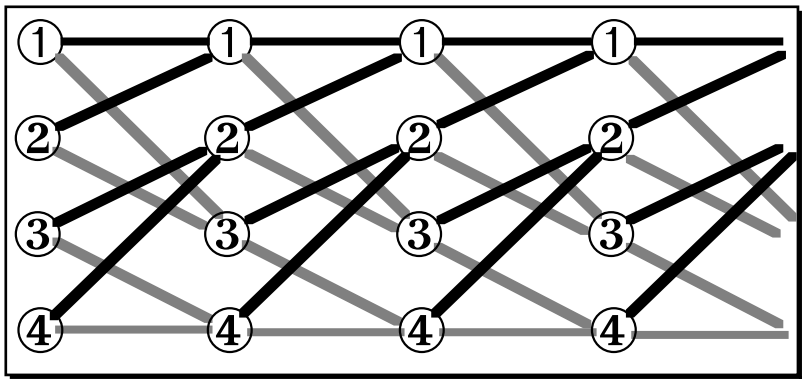
Mahdolliset tilat ovat $S_1 = \langle 0,0 \rangle$, $S_2 = \langle 0,1 \rangle$, $S_3 = \langle 1,0 \rangle$ ja $S_4 = \langle 1,1 \rangle$. Tiladiagrammi on ohessa.



HUOM! Automaattiteoriassa tällaisia koodereita kutsuttaisiin äärellisiksi jonokoneiksi eli transduktoreiksi, vrt. kurssi 73118 Formaaliset kielet tai MCELIECE & ASH & ASH.

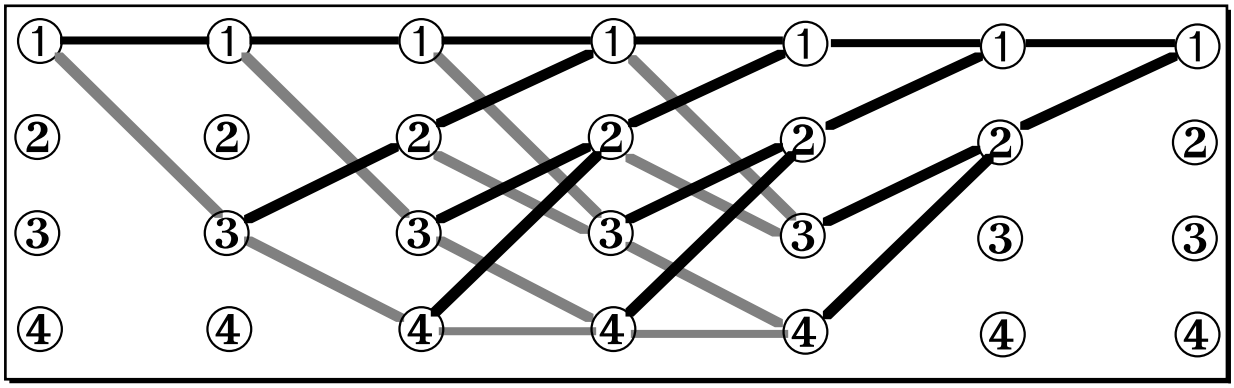
Toinen tapa esittää tilasiirtymät graafisesti on ns. *trellisdiagrammi.* Siinä piirretään tilat toistuvasti pystysuuntaan, jolloin vaakasuuntaan vasemmalta oikealle voidaan ilmaista tilojen vaihtelu ajan kuluessa. Trelliksestä voidaan löytää syötejonoa vastaavat tilareitit.

ESIMERKKI. (Jatkoa) Annettua tiladiagrammia vastaava trellisdiagrammi on oheisenlainen. Musta viiva edustaa siirtymää syötteellä 0 ja harmaa viiva syötteellä 1. Diagrammia voidaan periaatteessa jatkaa oikealle loputtomiin.



Alkutilaksi valitaan nyt $\langle \mathbf{0},\mathbf{0},\ldots,\mathbf{0}\rangle$. Koska käytännössä viestit eivät tietenkään ole äärettömän pitkiä, lopetetaan ne ajamalla kooderi jälleen tilaan $\langle \mathbf{0},\mathbf{0},\ldots,\mathbf{0}\rangle$. Tämä johtaa ns. *typistettyyn trellisdiagrammiin*.

ESIMERKKI. (Jatkoa) Eräs typistetty trellisdiagrammi on seuraava:



Paitsi kääntyvyyttä, oletetaan konvoluutiokooderilta, että se ei ole *katastrofinen*, ts. syötejono, jossa on äärettömän monta **0**:sta eroavaa termiä, ei saa johtaa koodattuun vasteeseen, jossa on vain äärellisen monta **0**:sta eroavaa termiä. (Jos näissä **0**:sta eroavissa paikoissa sattuisi sopiva virhe, palauttaisi dekoodaus pelkkiä **0**:ia sisältävän jonon ja näin tapahtuisi äärettömän monta dekoodausvirhettä.) Yleisen (n,k)-konvoluutiokooderin katastrofisuus on hankalahko karakterisoida (ks. HEISE & QUATTROCCHI). Polynomiaalisille (n,1)-konvoluutiokooderille saadaan

LAUSE 29. Polynomiaalinen (n,1)-konvoluutiokooderi, jonka siirtomatriisi on $\mathbf{H}(z) = (\mathbf{h}_1(z),\ldots,\mathbf{h}_n(z))$, on katastrofinen tarkalleen silloin, kun polynomeilla $\mathbf{h}_1(z),\ldots,\mathbf{h}_n(z)$ on yhteinen tekijä, joka ei ole muotoa fz^m, missä $f \in \mathbb{F}$ ja $m \geq 0$.

Todistus. Oletetaan ensin, että mainitunlainen yhteinen tekijä on olemassa. Luonnollisesti voidaan lisäksi olettaa, että tämän tekijän $\mathbf{p}(z)$ vakiotermi on $\overline{1}$, ts. $\mathbf{p}(z) = \overline{1} - z\mathbf{q}(z)$. Kehittämällä

$$\frac{\overline{1}}{\mathbf{p}(z)} = \frac{\overline{1}}{\overline{1} - z\mathbf{q}(z)}$$

formaalisesti ($z\mathbf{q}(z)$:n suhteen geometriseksi) sarjaksi $\mathbf{a}(z)$ saadaan sellaisen syötejonon z-muunnos, jossa on äärettömän monta $\overline{0}$:sta eroavaa termiä. Mutta aivan ilmeisesti $\mathbf{a}(z)\mathbf{H}(z)$ on polynomi, joten kooderi on katastrofinen.

Oletetaan sitten, että kooderi on katastrofinen ja merkitään polynomien $\mathbf{h}_1(z),\ldots,\mathbf{h}_n(z)$ suurinta yhteistä tekijää $\mathbf{d}(z)$:lla. ($\mathbf{d}(z)$ on siis korkeinta mahdollista astetta oleva polynomi, joka on kaikkien polynomien $\mathbf{h}_1(z),\ldots,$

$\mathbf{h}_n(z)$ tekijä.) Silloin (vrt. **III.1** ja totea, että yleisesti $\mathrm{syt}(\mathbf{c}_1(z),\ldots,\mathbf{c}_n(z)) = \mathrm{syt}(\mathbf{c}_1(z),\mathrm{syt}(\mathbf{c}_2(z),\ldots,\mathbf{c}_n(z)))$) voidaan kirjoittaa

$$\mathbf{d}(z) = \sum_{j=1}^{n} \mathbf{e}_j(z)\mathbf{h}_j(z).$$

Jos nyt $\mathbf{a}(z)$ on sellainen syötejonon z-muunnos, joka ei ole polynomi ja jota vastaavan vasteen z-muunnos $\mathbf{a}(z)\mathbf{H}(z) = (\mathbf{a}(z)\mathbf{h}_1(z),\ldots,\mathbf{a}(z)\mathbf{h}_n(z))$ on polynomialkioinen, niin

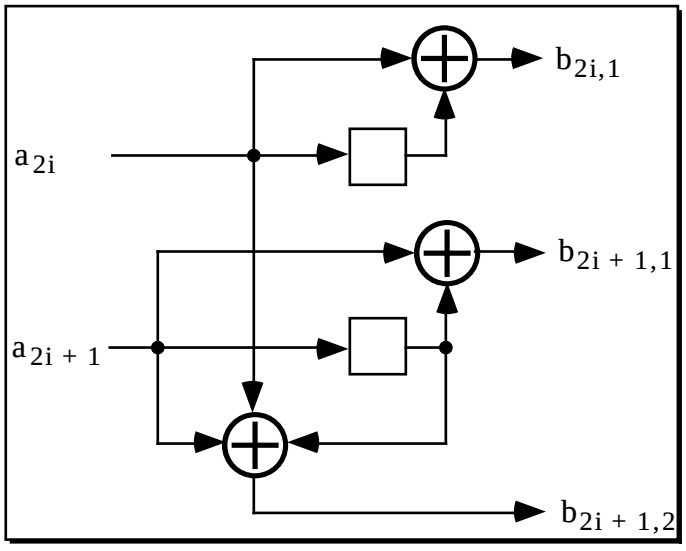
$$\mathbf{a}(z)\mathbf{d}(z) = \sum_{j=1}^{n} \mathbf{e}_j(z)\Big(\mathbf{a}(z)\mathbf{h}_j(z)\Big)$$

on polynomi. Tämä ei onnistu, jos $\mathbf{d}(z)$ on muotoa fz^m. ■

Konvoluutiokooderin ns. *koodaussuhde* on $\rho = k/n$ (vrt. lohkokoodien koodaussuhde **V.1**:ssä).

Ns. *puhkaistu* konvoluutiokoodaus saadaan, kun "tavallisen" konvoluutiokoodauksen koodatusta vastejonosta kietomalla saadusta jonosta poistetaan säännöllisin välein symboleita. Puhkaiseminen kasvattaa koodaussuhdetta. Luonnollisesti tällaista puhkaistua konvoluutiokoodausta vastaa toisella tavalla toteutettu "tavallinen" konvoluutiokoodaus, mutta sen dekoodaus on usein mutkikkaampaa.

ESIMERKKI. (Jatkoa) Puhkaistaan jättämällä systemaattisesti joka neljäs symboli pois, kietomalla esimerkiksi parillisilla i:n arvoilla mukaan vain b_{i1} ja parittomilla sekä b_{i1} että b_{i2}. Koodaussuhde kasvaa tällöin arvosta 1/2 arvoon 2/3. Vastaavan "tavallisen" kooderin siirtorekisteri on alla.



Koodatulle vastejonolle $[\ldots,\mathbf{0},\mathbf{0},\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2,\ldots]$ määritellään *(Hamming-)paino* luonnollisella tavalla:

$$w([\ldots,\mathbf{0},\mathbf{0},\mathbf{b}_0,\mathbf{b}_1,\mathbf{b}_2,\ldots]) = \sum_{i=0}^{\infty} w(\mathbf{b}_i).$$

Lohkokoodien minimietäisyyttä vastaa konvoluutiokoodauksessa *vapaa etäisyys* d_{free}, joka on pienin esiintyvä positiivinen koodatun vasteen paino. Polynomiaaliselle konvoluutiokooderille tämä on aina äärellinen (impulssivaste on polynomialkioinen).

Puhkaisussa vapaa etäisyys (yleensä) pienenee ja dekoodaus nopeutuu. Puhkaisuilla voidaankin toteuttaa samalla konvoluutiokooderilla useita eri vapaita etäisyyksiä. Jos virheitä on vähän, käytetään vahvasti puhkaistua koodausta ja vasta virheiden lukumäärän kasvaessa vähennetään puhkaisua ja lopulta käytetään täyttä koodausta.

Hyvien konvoluutiokooderien löytäminen on vaikeaa. Parhaat kooderit on löydetty tietokone-etsinnällä. Alla on taulukoitu (lähde: ANDERSON & MOHAN) eräitä tällä tavoin löydettyjä binäärisiä (n,1)-koodereita. Siirtomatriisin alkiot on annettu oktaalimuodossa (ks. s. 31).

n	k	d_{free}	siirtomatriisin alkiot
2	1	5	5,7
		6	15,17
		7	23,35
		8	65,57
		10	133,171
		10	345,237
		12	561,753
		12	1167,1545
		14	2335,3661
3	1	8	5,7,7
		10	13,15,17
		12	25,33,37
		13	47,53,75
		15	133,145,175
		16	113,155,173
		18	557,663,711
		20	1117,1365,1633
		22	2353,2671,3175
4	1	10	5,7,7,7
		13	13,15,15,17
		16	25,27,33,37
		18	53,67,71,75
		20	135,135,147,163
		22	235,275,313,357
		24	463,535,733,745
		27	1117,1257,1633,1653
		29	2327,2313,2671,3175

Seuraavassa on taulukoituna (lähde: ANDERSON & MOHAN) eräitä hyviä puhkaistuja binäärisiä konvoluutiokoodereita (alunperin n = 2 ja k = 1). Siirtomatriisin polynomit on annettu oktaalimuodossa, X on puhkaisukohta.

$\rho = 2/3$

d_{free}	siirtomatriisi
3	7,5,7,X
4	15,13,15,X
5	31,33,31,X
6	73,41,73,X
6	163,135,163,X
8	337,251,337,X
8	661,473,661,X

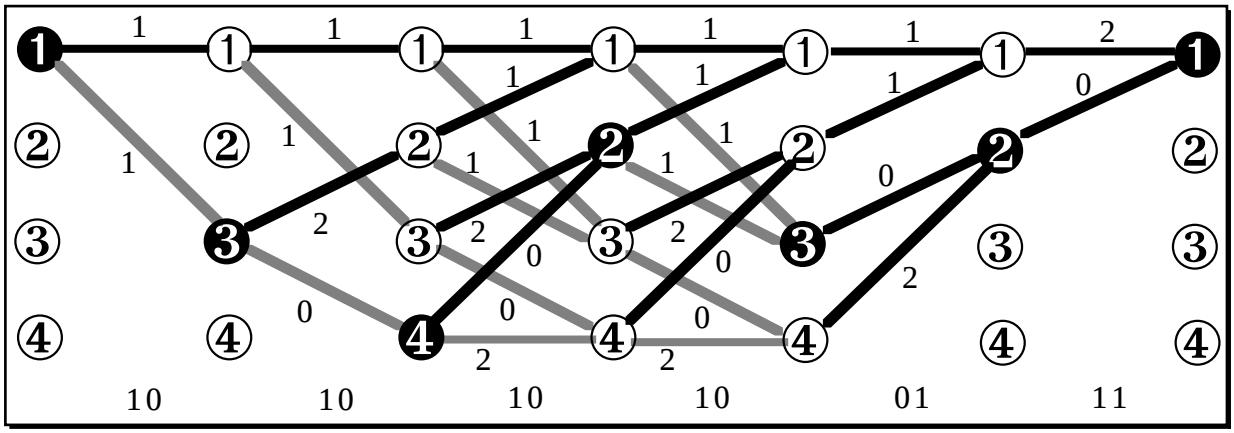
$\rho = 3/4$

d_{free}	siirtomatriisi
3	5,7,5,X,X,7
4	15,17,15,X,X,17
4	25,37,X,37,X,37
5	61,53,X,53,X,53
6	135,163,X,163,X,163
6	205,307,X,307,X,307
6	515,737,X,737,X,737

3. Viterbi-dekoodaus

Jos $[\mathbf{r}_1,\ldots,\mathbf{r}_P]$ on koodattu vastejono, johon on tullut mukaan virheitä, niin otetaan vastaava typistetty trellisdiagrammi ja yritetään löytää siitä mahdollisimman hyvin tähän jonoon sopiva tilasiirtymäketju. Tätä varten lasketaan ensin kullekin trelliksen viivoista paino seuraavasti: Jos kyseessä on tilasiirtymäketjun j:s viiva ja viivaa vastaava vastevektori on **b**, niin viivan paino on $d(\mathbf{r}_j,\mathbf{b})$. Näin trelliksestä tulee viivapainotettu graafi ja dekoodaus on kevyimmän polun etsimistä alkutilasta lopputilaan.

ESIMERKKI. (Jatkoa) Jos P = 6 ja virheellinen vastejono on [10,10,10,01,11], niin typistetyn trelliksen viivat painotetaan seuraavasti:



Kevyin polku on merkitty mustilla tilaympyröillä. Vastaava koodattu vastejono on [11,10,10,00,01,11]. Alkuperäinen viesti on 1101. (00 tarvitaan vielä ajamaan kooderi tilaan S_1.)

Koska dekoodaus on kevyimmän polun etsimistä graafista, mikä tahansa tähän tarkoitukseen tehdyistä algoritmeista kelpaa. *Viterbi-dekoodauksessa* käytetään ns. Dijkstran algoritmia*. Typistetyn trellisdiagrammin erikoisesta muodosta on tällöin tiettyä pientä etua aivan yleiseen graafiin verrattuna. Otetaan käyttöön seuraavat merkinnät:

$\mu_i(u)$ = tilan S_u (mahdollinen) paino typistetyn trelliksen i:nnessä sarakkeessa,

$\beta(u,v)$ = tilasiirtymään $S_u \rightarrow S_v$ liittyvä syötevektori,

$\rho_i(u,v)$ = typistetyn trelliksen i:nnessä ja i + 1:nnessä sarakkeessa olevien tilojen S_u ja S_v välisen viivan paino.

Näillä merkinnöin Viterbi-dekoodaus on seuraavanlainen menettely:

(1) Asetetaan $i \leftarrow 1$ ja $\mu_1(1) \leftarrow 0$, $W_1(1) \leftarrow []$ (tyhjä jono).

(2) Jos typistetyn trelliksen i + 1:nnen sarakkeen tilalla S_v on edeltäjätiloja i:nnessä sarakkeessa, niin valitaan niistä (jokin) sellainen tila S_{w_v}, että $\mu_i(w_v) + \rho_i(w_v,v)$ on pienin mahdollinen ja asetetaan

$$W_{i+1}(v) \leftarrow W_i(w_v),\beta(w_v,v),$$

$$\mu_{i+1}(v) \leftarrow \mu_i(w_v) + \rho_i(w_v,v)$$

($v = 1,\ldots,N$).

(3) Jos $i < P - 1$, asetetaan $i \leftarrow i + 1$ ja mennään kohtaan **(2)**. Muutoin tulostetaan $P - M$ ensimmäistä jonon $W_P(1)$ termiä ja lopetetaan.

Menetelmän haittana on suuri muistitilan tarve, kun $N = q^{kM}$ (lähinnä M) on suuri. Sen sijaan isotkin P:n arvot käyvät. Viterbi-dekoodaus sopii hyvin myös puhkaistujen koodauksien dekoodaukseen.

4. Jonodekoodaus

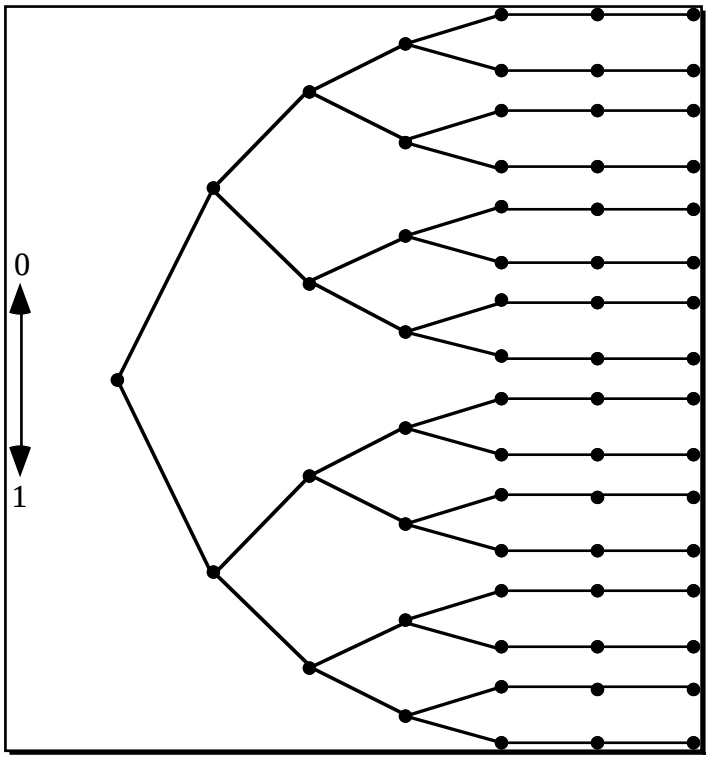
Jonodekoodausmenetelmissä pyritään seuraamaan mahdollisimman tarkasti syötejonon kulkua trelliksen sijasta ns. *koodauspuussa*. Puu haarau-

---

* Viterbi keksi kyllä algoritminsa Dijkstrasta riippumattomasti. Toinen asiaan läheisesti liittyvä algoritmi on Primin algoritmi. Ks. kurssi 73119 Graafiteoria tai MCELIECE & ASH & ASH. Itse asiassa nämä kaikki ovat dynaamisen optimoinnin sovelluksia (ks. kurssi 73125 Matemaattinen optimointiteoria 2).

tuu uudessa pisteessä aina q^k haaraan vastaten q^k erilaista mahdollista syötevektoria. Kutakin puun pistettä vastaa näin yksikäsitteinen kooderin tila sekä kutakin oksaa yksikäsitteinen syötevektori. Puun juuri vastaa alkutilaa $\langle \mathbf{0},\ldots,\mathbf{0}\rangle$. Kuten trelliksenkin tapauksessa puu typistetään ajamalla kooderi takaisin tilaan $\langle \mathbf{0},\ldots,\mathbf{0}\rangle$. Koska viimeiset M syötevektoria ovat näin aina nollavektoreita, ei puu enää tällöin haaraudu. Kutakin puun pistettä v vastaa yksikäsitteinen syötejono $A(v) = [\mathbf{a}_1,\ldots,\mathbf{a}_i]$, jolla siirrytään puun juuresta ko. pisteeseen. Samoin määräytyy yksikäsitteinen vastejono $B(v) = [\mathbf{b}_1,\ldots,\mathbf{b}_i]$ sekä yksikäsitteinen tilajono $S(v) = [S_{j_1},\ldots,S_{j_i}]$. Puun viimeiset pisteet ovat sen lehdet.

ESIMERKKI. (Jatkoa) Muodostetaan eo. typistettyä trellisdiagrammia vastaava typistetty koodauspuu.



Jos $[\mathbf{r}_1,\ldots,\mathbf{r}_P]$ on koodattu vastejono, johon on tullut mukaan virheitä, niin otetaan vastaava typistetty koodauspuu ja yritetään löytää siitä mahdollisimman hyvin tähän jonoon sopiva juuresta lähtevä polku. Tätä varten lasketaan ensin kullekin puun oksista paino seuraavasti: Jos kyseessä on polun i:s oksa ja viivaa vastaava tulostevektori on $\mathbf{b}_i$, niin oksan paino on $d(\mathbf{r}_i,\mathbf{b}_i)$. Edelleen polun p paino $w(p)$ on sen oksien painojen summa. Kutakin puun pistettä v vastaa näin yksikäsitteinen siihen päättyvän polun paino.

HUOM! Eo. polun paino on Hamming-paino. Jos tiedetään eri virheiden esiintymistodennäköisyydet, voidaan ottaa käyttöön yleisempikin paino,

ns. Fano-paino, ks. esimerkiksi MCELIECE. Hamming-paino vastaa tilannetta, jossa kaikki virheet esiintyvät yhtä todennäköisesti.

Jonodekoodausalgoritmeissa tarvittavia painoja yms. ei lasketa kaikkia kerralla, vaan sitä mukaa kun niitä tarvitaan.

m-algoritmi

m-algoritmissa ylläpidetään tiedostoa, jossa on m toistaiseksi parasta löytynyttä polkua. Kun on löytynyt polku juuresta lehteen, lopetetaan.

(1) Asetaan $X \leftarrow \{0\}$ (0 on tyhjä polku, jonka alku- ja loppupiste on koodauspuun juuri).

(2) Jos X:ssä on polkuja juuresta lehteen v, valitaan näistä paras (eli painoltaan kevyin), tulostetaan sitä vastaavan A(v):n P – M ensimmäistä termiä ja lopetetaan.

(3) Jos X:ssä ei vielä ole polkuja juuresta lehteen, jatketaan kutakin X:ssä olevaa polkua yhdellä oksalla kaikilla mahdollisilla tavoilla ja lasketaan näille painot.

(4) Valitaan näin saaduista poluista m parasta $p_1,\ldots,p_m$ (eli painoltaan kevyintä) asetetaan $X \leftarrow \{p_1,\ldots,p_m\}$ ja mennään kohtaan **(2)**.

m-algoritmi on yksinkertaisesti toteutettavissa, mutta “hukkaa” helposti oikean polun kohdassa **(4)**. Tämän estämiseksi pitää valita suuri m:n arvo, mikä taas johtaa suureen muistitilan tarpeeseen ja pitkään lajitteluaikaan.

Pinoalgoritmi

Pinoalgoritmissa ylläpidetään pinoa (oikeastaan jonoa), jossa ovat kaikki kulloinkin tutkitut polut paremmuusjärjestyksessä, paras ylimpänä. Kun pinon päällimmäisenä on polku juuresta lehteen, lopetetaan.

(1) Asetaan $Y \leftarrow [0)$ (0 on tyhjä polku, jonka alku- ja loppupiste on koodauspuun juuri).

(2) Jos Y:ssä on päällimmäisenä polku juuresta lehteen v, tulostetaan sitä vastaavan A(v):n P – M ensimmäistä termiä ja lopetetaan.

(3) Jos Y:ssä ei ole päällimmäisenä polkua juuresta lehteen, jatketaan päällimmäisenä olevaa polkua yhdellä oksalla kaikilla

mahdollisilla tavoilla, lasketaan näin saatujen polkujen painot ja sijoitetaan ne oikeisiin paikkoihinsa pinossa. Asetetaan Y:lle arvoksi näin saatu uusi pino. Mennään kohtaan **(2)**.

Pinoalgoritmin haittapuolena on pinon kasvaminen ajoittain suureksi, jolloin sen ylläpitämiseen kuluu paljon muistitilaa ja lajitteluun kuluu paljon aikaa. Käytännössä pinolle pitää asettaa maksimikorkeus ja lopettaa algoritmin suoritus, kun pino saavuttaa tämän korkeuden. Tällöin jää dekoodaus kokonaan tekemättä!

Monipinoalgoritmi

Monipinoalgoritmissa muodostetaan ensin pääpino Y_0 samalla tavoin, kun yllä, ja jos tämä pino täyttyy maksimikorkeuteensa K_0 ilman että saataisiin dekoodattua viestiä, otetaan pinosta T polkua "siemeneksi" uuteen pinoon Y_1. Mikäli Y_1 saavuttaa maksimikorkeuden K, perustetaan jälleen uusi pino Y_2, jne.. Yleensä K on paljon pienempi kuin K_0 ja samoin T on pieni luku.

(1) Asetetaan $i \leftarrow 0$ ja käydään läpi "tavallinen" pinoalgoritmi pinossa Y_0, jonka maksimikorkeus on K_0. Jos tämä tuottaa tuloksen, tulostetaan vastaava syötejono ja lopetetaan. Muutoin Y_0 täyttyy ja asetetaan $\omega \leftarrow \infty$.

(2) **(2.1)** Merkitään Y_i:n polut, joiden paino on $>\omega$. Jos Y_i:ssä on nyt T merkitsemätöntä polkua $p_1,\ldots,p_T$ (paremmuusjärjestyksessä), merkitään vielä nämä polut ja asetetaan $i \leftarrow i + 1$ sekä $Y_i \leftarrow [p_1,\ldots,p_T)$. Mennään kohtaan **(3)**.

(2.2) Jos merkitsemättömiä polkuja ei ole tarpeeksi ja $i \geq 1$, poistetaan pino Y_i, asetetaan $i \leftarrow i - 1$ ja palataan kohtaan **(2)**.

(2.3) Jos merkitsemättömiä polkuja ei ole tarpeeksi ja $i = 0$, tulostetaan W:n P – M ensimmäistä termiä ja lopetetaan.

(3) Käydään läpi "tavallinen" pinoalgoritmi pinossa Y_i, jonka maksimikorkeus on K.

(3.1) Jos tämä tuottaa tuloksena polun p juuresta lehteen v, niin poistetaan pino Y_i ja asetetaan $i \leftarrow i - 1$. Jos vielä $w(p) < \omega$, asetetaan $W \leftarrow A(v)$ ja $\omega \leftarrow w(p)$. Mennään kohtaan **(2)**.

(3.2) Jos taas pino Y_i täyttyy, mennään kohtaan **(2)**.

Käytännössä kaikkien yhtaikaa käytössä olevien pinojen yhteiskorkeudelle pitää asettaa maksimi ja lopettaa, kun tämä saavutetaan. Jos tällöin $\omega < \infty$ eli on löytynyt polku p juuresta lehteen, tulostetaan W:n P – M ensimmäistä termiä. Jos $\omega = \infty$, ei dekoodausta voitu suorittaa lainkaan! Monipinoalgoritmin etuna verrattuna "tavalliseen" pinoalgoritmiin on vähäisempi muistitilan tarve ja matalien lisäpinojen nopea lajiteltavuus.

Fano-dekoodaus

Fano-dekoodauksessa verrataan kulloinkin tutkittavan polun p painoa tiettyyn kynnysarvoon σ ja polkua jatketaan niin kauan kuin kynnysarvo ei ylity. Kynnysarvon ylityttyä palataan polkua takaisin yhden oksan verran ja yritetään jatkaa sitä jotain toista oksaa pitkin, jne.. Jos tämäkään ei onnistu, nostetaan lopulta kynnysarvoa tietyllä lisäyksellä δ ja jatketaan, kunnes polku juuresta lehteen on löytynyt. Merkitään S(p):llä kaikkia polusta p yhdellä oksalla jatkamalla saatujen polkujen joukkoa ja N(p):llä kaikkien polusta p viimeisen oksan osalta eroavien polkujen joukkoa. I(p):llä taas merkitään polkua, joka saadaan poistamalla p:stä viimeinen oksa. Jotta saataisiin S(p):n ja N(p):n polut järjestettyä "paremmuusjärjestykseen", menetellään seuraavasti: Määritellään ensin järjestys kaikille mahdollisille syötevektoreille. Tämä järjestäminen on mielivaltainen, mutta sen pitäisi olla laskennallisesti nopeasti testattavissa. Määritellään sitten järjestys $\prec$ poluille: $p \prec q$, jos joko $w(p) < w(q)$ tai sitten on $w(p) = w(q)$ ja p:n viimeistä oksaa vastaava syötevektori on järjestyksessä pienempi kuin q:n vastaava. Huomaa, että $\prec$ todella järjestää S(p):n ja N(p):n polut paremmuusjonoon.

(1) Asetetaan $\sigma \leftarrow 0$ ja $p \leftarrow o$ (tyhjä polku, jonka alku- ja loppupiste on koodauspuun juuri).

(2) Jos p on polku juuresta lehteen v, tulostetaan A(v):n P – M ensimmäistä termiä ja lopetetaan. Muussa tapauksessa muodostetaan S(p):n polut ja valitaan niistä paras polku q ($\prec$:n suhteen).

(3) Jos $w(q) \leq \sigma$, asetetaan $p \leftarrow q$ ja mennään kohtaan **(2)**.

(4) Jos $p = o$, asetetaan $\sigma \leftarrow \sigma + \delta$ ja mennään kohtaan **(2)**. Muussa tapauksessa muodostetaan N(p):n polut.

(5) **(5.1)** Mikäli N(p):ssä on polkuja, joiden paino ei ylitä kynnysarvoa σ, valitaan näistä poluista paras sellainen polku r, joka toteuttaa ehdon $p \prec r$, asetetaan $p \leftarrow r$ ja mennään kohtaan **(2)**. (Huomaa, miten vaatimus $p \prec r$ estää jo kokeiltujen polkujen uudelleen valitsemisen.)

(5.2) Jos taas kaikkien ehdon $p \prec r$ toteuttavien N(p):n polkujen r painot ylittävät kynnysarvon σ, asetetaan $p \leftarrow$ I(p) ja mennään kohtaan **(4)**.

Fano-dekoodaus käyttää muistitilaa minimaalisen vähän. Sen sijaan siltä saattaa kulua hyvinkin paljon aikaa koodauspuussa “kulkemiseen”. (Ilmankos HEISE & QUATTROCCHI käyttääkin nimeä “Decodiereraffe”.)

Etsintämielessä Viterbi-dekoodaus sekä m-algoritmi ovat “breadth–first-tyyppisiä”, pinoalgoritmi on “best–first-tyyppinen” ja Fano-dekoodaus taas “depth–first-tyyppinen” (ks. kurssi 73119 Graafiteoria). Monipinoalgoritmi on “best–first-depth–first-hybridi”.

Lähemmin konvoluutiokoodauksen dekoodausta käsitellään viitteissä ANDERSON & MOHAN, LIN & COSTELLO, MCELIECE ja VITERBI & OMURA.

VII LUKU

INFORMAATIO JA ENTROPIA

1. Ekskursio: Diskreetti todennäköisyys

Kerrataan joitakin tilastomatematiikan kurssin käsitteitä ja otetaan käyttöön uusia merkintöjä.

Jos A ja B ovat tapauksia, niin

1) $A \cup B$ = {A tai/ja B tapahtuu};

2) $A \cap B$ = {A ja B molemmat tapahtuvat};

3) A ja B ovat *erilliset*, jos $P(A \cap B) = 0$;

4) $P(A \cup B) = P(A) + P(B) - P(A \cap B)$;

5) A:n todennäköisyys, kun B:n tiedetään tapahtuvan, (*ehdollinen todennäköisyys*) on

$$\frac{P(A \cap B)}{P(B)} = P(A \mid B).$$

6) Jos siis $P(B) \neq 0$, niin $P(A \cap B) = P(A \mid B)P(B)$. Kaava yleistyy, sillä

$$P(A_1 \cap \cdots \cap A_n) = P(A_1 \mid A_2 \cap \cdots \cap A_n)P(A_2 \cap \cdots \cap A_n)$$

$$= P(A_1 \mid A_2 \cap \cdots \cap A_n)P(A_2 \mid A_3 \cap \cdots \cap A_n)P(A_3 \cap \cdots \cap A_n)$$

$$= \cdots = P(A_1 \mid A_2 \cap \cdots \cap A_n)P(A_2 \mid A_3 \cap \cdots \cap A_n) \cdots P(A_{n-1} \mid A_n)P(A_n).$$

7) Tapaukset A ja B ovat *riippumattomat*, jos

$$P(A \cap B) = P(A)P(B).$$

Muussa tapauksessa ne ovat *riippuvat*. Jos A ja B ovat riippumattomat ja $P(B) \neq 0$, niin $P(A \mid B) = P(A)$.

SUURTEN LUKUJEN LAKI. Toistetaan (stokastista) koetta, jossa suotuisan tapauksen todennäköisyys on p. Merkitään S_n:llä suotuisien tapausten lukumäärää n koetoistossa (toistot ovat riippumattomat). Kaikille positiiviluvuille ε ja δ (miten tahansa pienille) on olemassa sellainen vakio $N_{\varepsilon,\delta}$, että

$$P\left(\left|\frac{S_n}{n} - p\right| < \varepsilon\right) > 1 - \delta, \text{ kun } n > N_{\varepsilon,\delta}. \quad \blacksquare$$

Tapaukset $A_1,\ldots,A_n$ muodostavat *täydellisen tapausjärjestelmän,* jos

(i) $P(A_1) + \cdots + P(A_n) = 1$ ja

(ii) $P(A_i \cap A_j) = 0$, kun $i \neq j$, ts. tapaukset ovat parittain erilliset.

KOKONAISTODENNÄKÖISYYSKAAVA. Jos $A_1,\ldots,A_n$ on täydellinen tapausjärjestelmä, niin

$$P(B) = \sum_{i=1}^{n} P(B \cap A_i) = \sum_{i=1}^{n} P(B|A_i)P(A_i). \quad \blacksquare$$

Jos S_1: $A_1,\ldots,A_n$ ja S_2: $B_1,\ldots,B_m$ ovat täydellisiä tapausjärjestelmiä, niin samoin on *yhteistapausjärjestelmä*

$$S_1 \cap S_2\text{: } A_i \cap B_j \quad (i = 1,\ldots,n; j = 1,\ldots,m).$$

Ensinnäkin Kokonaistodennäköisyyskaavan nojalla

$$\sum_{i=1}^{n}\sum_{j=1}^{m} P(A_i \cap B_j) = \sum_{i=1}^{n} P(A_i) = 1,$$

ja toiseksi, jos $i_1 \neq i_2$ tai/ja $j_1 \neq j_2$, niin

$$P((A_{i_1} \cap B_{j_1}) \cap (A_{i_2} \cap B_{j_2})) = P((A_{i_1} \cap A_{i_2}) \cap (B_{j_1} \cap B_{j_2})) = 0.$$

Olettaen, että $P(B_1),\ldots,P(B_m) \neq 0$, saadaan *ehdollinen tapausjärjestelmä*

$$S_1|S_2 : A_i|B_j \quad (i = 1,\ldots,n; j = 1,\ldots,m).$$

Tämä ei ole täydellinen tapausjärjestelmä, vaan oikeastaan m:n täydellisen tapausjärjestelmän

$$S_1|B_j : A_1|B_j,\ldots,A_n|B_j \qquad (j = 1,\ldots,m)$$

kokoelma. Vastaavasti saataisiin $S_2|S_1$.

Laskettaessa (esimerkiksi Matlab-ohjelmistolla) on kätevää siirtyä vektori/matriisinotaatioon. Täydellisen tapausjärjestelmän S_1: $A_1,\ldots,A_n$ todennäköisyydet

$$P(A_1) = p_1 \ ,\ldots,\ P(A_n) = p_n$$

kirjoitetaan tapausjärjestelmän *todennäköisyysvektoriksi*

$$\mathbf{p} = (p_1,\ldots,p_n).$$

Vastaavasti kahden tapausjärjestelmän S_1: $A_1,\ldots,A_n$ ja S_2: $B_1,\ldots,B_m$ yhteistapausjärjestelmän $S_1 \cap S_2$ todennäköisyydet

$$P(A_i \cap B_j) = p_{ij} \quad (i = 1,\ldots,n; j = 1,\ldots,m)$$

kirjoitetaan *todennäköisyysmatriisiksi*

$$\mathbf{P} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nm} \end{pmatrix}.$$

Ehdollisen tapausjärjestelmän todennäköisyydet

$$P(A_i \mid B_j) = p_{i\mid j} \quad (i = 1,\ldots,n; j = 1,\ldots,m)$$

voidaan myös kirjoittaa ns. *ehdolliseksi todennäköisyysmatriisiksi*

$$\mathbf{Q} = \begin{pmatrix} p_{1\mid 1} & \cdots & p_{1\mid m} \\ \vdots & \ddots & \vdots \\ p_{n\mid 1} & \cdots & p_{n\mid m} \end{pmatrix}.$$

Jos otetaan käyttöön myös S_2:n todennäköisyysvektori

$$\mathbf{q} = (q_1,\ldots,q_m),$$

missä

$$P(B_1) = q_1 \ ,\ldots,\ P(B_m) = q_m,$$

sekä ykkösvektori

$$\mathbf{1}_n = (\underbrace{1,\ldots,1}_{n \text{ kpl}}),$$

niin (Kokonaistodennäköisyyskaavan nojalla) saadaan seuraavat yhtälöt:

$$\mathbf{p} = \mathbf{1}_m\mathbf{P}^T, \quad \mathbf{q} = \mathbf{1}_n\mathbf{P},$$

$$\mathbf{p} = \mathbf{q}\mathbf{Q}^T, \qquad \mathbf{P} = \mathbf{Q}\lceil\mathbf{q}\rfloor,$$

$$\mathbf{Q} = \mathbf{P}\lceil \mathbf{q} \rfloor^{-1}$$

(tässä $\lceil \mathbf{q} \rfloor$ on lävistäjämatriisi, joka saadaan ottamalla lävistäjäalkioiksi **q**:n komponentit). Jos **p** ja **Q** tunnetaan, mutta ei **q**:ta (eikä **P**:tä), niin se voidaan ratkaista (tai ainakin löytää jokin **q**) yhtälöstä $\mathbf{p} = \mathbf{q}\mathbf{Q}^T$. Huomaa, että tällöin automaattisesti $\mathbf{1}_m\mathbf{q}^T = \mathbf{1}_n\mathbf{Q}\mathbf{q}^T = \mathbf{1}_n\mathbf{p}^T = 1$.

Lasketaan Matlabilla esimerkki:

```
P =
    0.0187    0.0001    0.0753    0.0222    0.0042
    0.0402    0.0185    0.0405    0.3014    0.0143
    0.0212    0.0667    0.0488    0.1739    0.0086
    0.0287    0.0201    0.0037    0.0806    0.0122

»p=sum(P')

p =
    0.1205    0.4149    0.3193    0.1453

»q=sum(P)

q =
    0.1088    0.1055    0.1683    0.5781    0.0393

»Q=P*inv(diag(q))

Q =
    0.1716    0.0009    0.4473    0.0384    0.1078
    0.3693    0.1758    0.2405    0.5213    0.3638
    0.1950    0.6329    0.2900    0.3009    0.2194
    0.2641    0.1904    0.0222    0.1394    0.3090
```

Jos **q** ei olisi tunnettu, voitaisiin yrittää etsiä se käyttäen $\mathbf{Q}^T$:n yleistettyä käänteismatriisia, ensin Moore–Penrose-inverssiä ja sitten pienimmän neliösumman inverssiä:

```
»p*pinv(Q')

ans =
    0.0937    0.1040    0.1708    0.5788    0.0527

»p/Q'

ans =
         0    0.0948    0.1865    0.5835    0.1351
```

2. Informaatio

Tapauksen A *(itse)informaatio* on

$$\boxed{-\log P(A) = I(A).}$$

Varman tapauksen informaatio on siis 0 ja mahdottoman tapauksen ∞. Logaritmin kantaluku määrää informaation yksikön. Mikä tahansa kantaluku $a > 1$ on mahdollinen. Tavallisimmat valinnat ovat

$a = 2$:	$I(A) = -\log_2 P(A)$	*bittiä*
$a = e$:	$I(A) = -\ln P(A)$	*nattia*
$a = 10$:	$I(A) = -\lg P(A)$	*dittiä*

Yksikön vaihto käy tavalliseen tapaan:

$$\frac{\log_2 x}{\log_2 a} = \frac{\ln x}{\ln a} = \frac{\lg x}{\lg a} = \log_a x.$$

3. Entropia

Täydellisen tapausjärjestelmän $S : A_1,\ldots,A_n$ *entropia* on

$$\boxed{H(S) = \sum_{i=1}^{n} I(A_i)P(A_i),}$$

ts. entropia on järjestelmän tapausten keskimääräinen informaatio. Entropian yksikkö on sama kuin käytetty informaation yksikkö. Tässä sovitaan, että $0 \log 0 = 0$. Tämä on sopusoinnussa sen peruskursseilta tutun seikan kanssa, että $\lim_{x \to 0+} x \ln x = 0$.

Jos $\mathbf{p} = (p_1,\ldots,p_n)$ on S:n todennäköisyysvektori, niin

$$\boxed{H(S) = -\sum_{i=1}^{n} p_i \log p_i = -\mathbf{p} \log \mathbf{p}^T,}$$

kun merkitään $\log \mathbf{p} = (\log p_1,\ldots,\log p_n)$. Vastaavasti, jos kyseessä on yhteistapausjärjestelmä $S_1 \cap S_2$, jonka todennäköisyysmatriisi on $\mathbf{P}$, niin

$$\boxed{H(S_1 \cap S_2) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij} = -\mathrm{trace}(\mathbf{P} \log \mathbf{P}^T),}$$

missä* on merkitty

$$\log \mathbf{P} = \begin{pmatrix} \log p_{11} & \cdots & \log p_{1m} \\ \vdots & \ddots & \vdots \\ \log p_{n1} & \cdots & \log p_{nm} \end{pmatrix}.$$

Esimerkiksi yo. Matlab-esimerkissä

```
»H=-p*log(p')/log(2)

H =
    1.8247

»HS12=-trace(P*log(P'))/log(2)

HS12 =
    3.3843
```

(yksikkönä bitti).

Merkitään $g(p) = -p \log p$. Silloin ilmeisesti $g(0) = g(1) = 0$ ja $g(p) > 0$, kun $0 < p < 1$. $H(S)$ on näin ollen ei–negatiivinen ja $H(S) = 0$ tarkalleen siinä tapauksessa, että yksi todennäköisyyksistä $p_1,\ldots,p_n$ on =1 muiden ollessa =0. g(p):n maksimiarvo $\frac{1}{e \ln 2}$ bittiä saavutetaan, kun $p = 1/e$. Edelleen voidaan todeta, että jos $p = p_1 + p_2$ ja $p_1, p_2 \neq 0$, niin (nateissa)

$$g(p) = -(p_1 + p_2) \ln p = p_1 \ln \frac{p_1}{p} + p_2 \ln \frac{p_2}{p} + g(p_1) + g(p_2)$$

$$< g(p_1) + g(p_2).$$

Piirretään vielä Maplella g(p):n kuvaaja (biteissä):

---

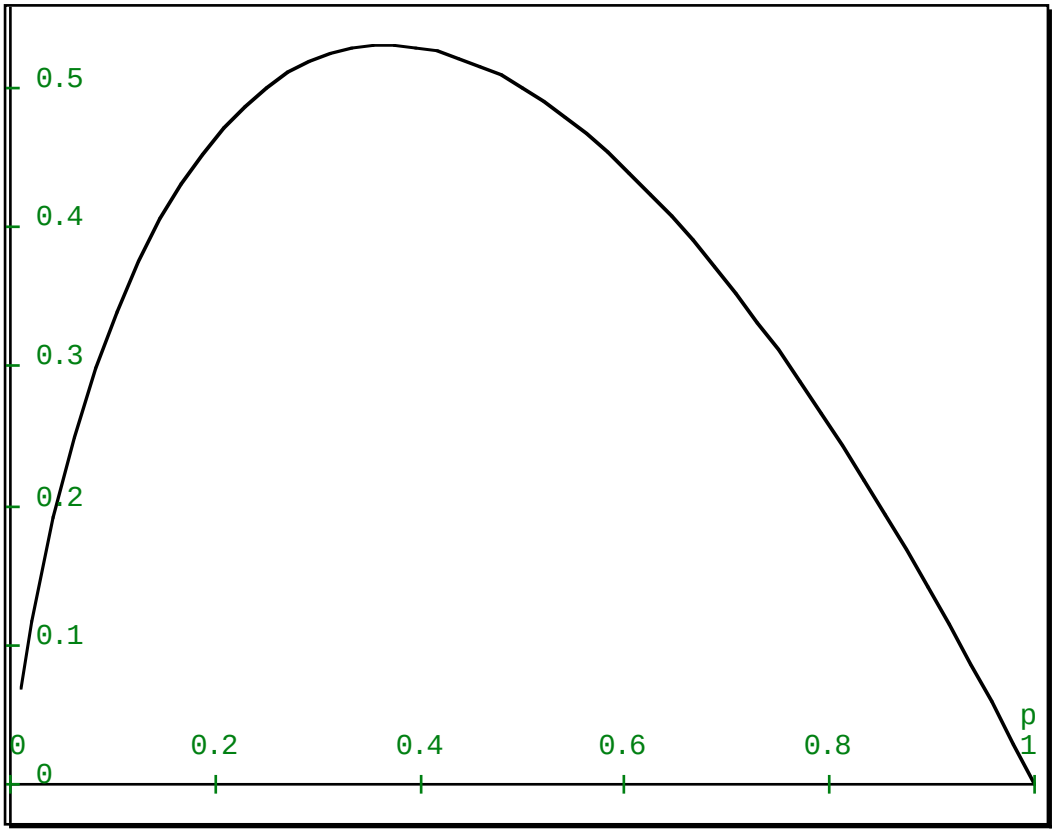
* Matriisin *trace* eli *jälki* on sen lävistäjäalkioiden summa. Seuraavat tracen ominaisuudet ovat helposti todettavissa:

(i) $\mathrm{trace}(\mathbf{A} + \mathbf{B}) = \mathrm{trace}(\mathbf{A}) + \mathrm{trace}(\mathbf{B})$, **(ii)** $\mathrm{trace}(\mathbf{A}^T) = \mathrm{trace}(\mathbf{A})$,

(iii) $\mathrm{trace}(c\mathbf{A}) = c\,\mathrm{trace}(\mathbf{A})$, kun c on vakio, **(iv)** $\mathrm{trace}(\mathbf{AB}) = \mathrm{trace}(\mathbf{BA})$,

(v) $\mathrm{trace}(\mathbf{AB}^T) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} b_{ij}$, kun $\mathbf{A} = (a_{ij})$ ja $\mathbf{B} = (b_{ij})$.

Ominaisuus **(v)** antaa matriisien skalaaritulon, jota nimenomaan käytetään laskettaessa entropiaa $H(S_1 \cap S_2)$.



APULAUSE. $\ln x \le x - 1$, kun $x > 0$. Yhtäsuuruus esiintyy vain x:n arvolla 1.

Todistus. Funktion $x - 1 - \ln x$ ainoa ääriarvo on pisteessä $x = 1$ oleva minimiarvo 0. ■

LAUSE 30. $H(S) \le \log n$. Yhtäsuuruus esiintyy vain tapauksessa $p_1 = \cdots = p_n$.

Todistus. Jos $p_i = 0$, niin

$$-p_i \ln n - p_i \ln p_i = 0 < \frac{1}{n} - p_i.$$

Jos taas $p_i \neq 0$, niin Apulauseen nojalla

$$-p_i \ln n - p_i \ln p_i = p_i \ln \frac{1}{n p_i} \le p_i\left(\frac{1}{n p_i} - 1\right) = \frac{1}{n} - p_i$$

ja yhtäsuuruus esiintyy vain jos $p_i = 1/n$. Näin ollen

$$\sum_{i=1}^{n}(-p_i \ln n - p_i \ln p_i) \le \sum_{i=1}^{n}\left(\frac{1}{n} - p_i\right) = \sum_{i=1}^{n}\frac{1}{n} - \sum_{i=1}^{n} p_i = 1 - 1 = 0$$

ja yhtäsuuruus esiintyy vain tapauksessa $p_1 = \cdots = p_n = 1/n$. Edelleen (nateissa)

$$H(S) = -\sum_{i=1}^{n} p_i \ln p_i \le \sum_{i=1}^{n} p_i \ln n = \ln n \sum_{i=1}^{n} p_i = \ln n$$

ja yhtäsuuruus pätee vain ym. tapauksessa. Muille informaatioyksiköille tulos saadaan kertomalla puolittain sopivalla vakiolla. ■

Entropia on maksimissaan kun kaikki mahdollisuudet (tapausjärjestelmä) ovat yhtä todennäköisiä, ts. (a priori) epävarmuus tuloksesta on suurin. Entropia on epävarmuuden mitta: mitä suurempi entropia sitä epävarmempaa on tulos. Jotta kokeesta saataisiin mahdollisimman paljon tulosta, se kannattaa asettaa siten, että epävarmuus kokeen tuloksesta on ennalta suuri (maksimientropiaperiaate, ks. Luku **X**). Mukana voi olla lisäehtoja, jolloin Lauseen **30** antamaa suurinta entropiaa ln n nattia ei voida saavuttaa. Tällöin entropia maksimoidaan annettujen lisäehtojen puitteissa.

Katsotaan paria entropian kasvattamiseen liittyvää asiaa. Valitaan täydelliset tapausjärjestelmät

$$S: A_1, A_2, A_3, \ldots, A_n \; ; \qquad S': A_1 \cup A_2, A_3, \ldots, A_n \quad \text{ja} \quad S'': B, C, A_3, \ldots, A_n \, ,$$

missä $p_1 = P(A_1) < P(B) < P(C) < P(A_2) = p_2$. Koska $g(p_1 + p_2) < g(p_1) + g(p_2)$, on $H(S) > H(S')$. Entropian kasvattamiseksi pitää siis aina jakaa tilanteet osiin mahdollisuuksien mukaan. Merkitään nyt $q_1 = P(B)$ ja $q_2 = P(C)$ sekä $q_1 - p_1 = w$, jolloin myös $p_2 - q_2 = w$ (miksi?). Silloin (nateissa)

$$\begin{aligned} g(p_1) + g(p_2) - g(q_1) - g(q_2) &= p_1 \ln \frac{q_1}{p_1} + p_2 \ln \frac{q_2}{p_2} + w \ln \frac{q_1}{q_2} \\ &\le p_1\left(\frac{q_1}{p_1} - 1\right) + p_2\left(\frac{q_2}{p_2} - 1\right) + w \ln \frac{q_1}{q_2} = w \ln \frac{q_1}{q_2} < 0, \end{aligned}$$

ts. $H(S'') > H(S)$. Jaettaessa tilanteita ym. tavalla osiin, kannattaa entropian maksimoimiseksi siis pyrkiä mahdollisimman tarkasti yhtä todennäköisiin osiin.

4. Ehdollinen entropia. Keskinäisinformaatio

Tapauksen A *ehdollinen informaatio* ehdolla, että B on tapahtunut, on

$$\boxed{-\log P(A \mid B)} = -\log \frac{p(A \cap B)}{P(B)} = -\log P(A \cap B) + \log P(B)$$

$$\boxed{= I(A \cap B) - I(B) \quad = I(A \mid B)}$$

(olettaen tietysti, että $P(B) \neq 0$).

Tarkastellaan kahta täydellistä tapausjärjestelmää

$$S_1 : A_1,\ldots,A_n \qquad \text{ja} \qquad S_2 : B_1,\ldots,B_m$$

ja yhteistapausjärjestelmää

$$S_1 \cap S_2 : A_i \cap B_j \quad (i = 1,\ldots,n; j = 1,\ldots,m).$$

S_1:n *ehdollinen entropia* ehdolla S_2 on

$$H(S_1|S_2) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ P(B_j)\neq 0}}^{m} I(A_i|B_j)P(A_i \cap B_j).$$

Ehdollinen entropia $H(S_1|S_2)$ on S_1:n tapausten keskimääräinen ehdollinen informaatio S_2:n tapauksilla ehdollistettuina. Odotusarvo (eli keskiarvo) otetaan yli yhteistapausjärjestelmän $S_1 \cap S_2$. $H(S_1|S_2)$ kuvaa näin ollen S_1:n keskimääräistä epävarmuutta, kun S_2:n tulos tiedetään.

Merkitään S_1:n ja S_2:n todennäköisyysvektoreita **p**:llä ja **q**:lla sekä yhteistapausjärjestelmän todennäköisyysmatriisia ja ehdollista todennäköisyysmatriisia **P**:llä ja **Q**:lla. Jos $P(B_j) = q_j \neq 0$, on silloin

$$P(A_i|B_j) = \frac{p_{ij}}{q_j}$$

ja

$$I(A_i|B_j) = -\log \frac{p_{ij}}{q_j}\,.$$

Mikäli nyt $p_{ij} = 0$ (mutta $q_j \neq 0$), on luonnollista sopia, että

$$p_{ij} \log \frac{p_{ij}}{q_j} = p_{ij} \log p_{ij} - p_{ij} \log q_j = 0.$$

Ulotetaan tämä sopimus myös tapaukseen $q_j = 0$ (jolloin myös $p_{ij} = 0$), ts. tällöinkin*

---

* Myös tämä on sopusoinnussa raja-arvotulosten kanssa. Jos nimittäin $P(B) \neq 0$, on

$$P(A \cap B) = P(A|B)P(B) \leq P(B).$$

Toisaalta, jos $(p,q) \to (0,0)$ ja $0 \leq p \leq q$, niin

$$0 \leq |p \ln q| = p|\ln q| \leq q|\ln q| = |q \ln q| \to 0.$$

$$p_{ij} \log \frac{p_{ij}}{q_j} = 0$$

ja tietysti myös $p_{ij} \log q_j = 0$. Näin ollen voidaan kirjoittaa

$$\boxed{H(S_1|S_2) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log \frac{p_{ij}}{q_j} = -\mathrm{trace}(\mathbf{P} \log \mathbf{Q}^{\mathrm{T}}).}$$

Tässä on merkitty

$$\log \mathbf{Q} = \begin{pmatrix} \log p_{1|1} & \cdots & \log p_{1|m} \\ \vdots & \ddots & \vdots \\ \log p_{n|1} & \cdots & \log p_{n|m} \end{pmatrix}.$$

Jatketaan eo. Matlab-esimerkkiä:

```
»HS1S2=-trace(P*log(Q'))/log(2)
HS1S2 =
    1.6205
```

LAUSE 31. $\boxed{H(S_1 | S_2) = H(S_1 \cap S_2) - H(S_2)}$

Todistus. Kokonaistodennäköisyyskaavan nojalla

$$H(S_1|S_2) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log \frac{p_{ij}}{q_j} = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij} + \sum_{j=1}^{m}\sum_{i=1}^{n} p_{ij} \log q_j$$

$$= -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij} + \sum_{j=1}^{m} q_j \log q_j = H(S_1 \cap S_2) - H(S_2). \quad \blacksquare$$

$H(S_1)$ mittaa S_1:n keskimääräistä epävarmuutta ennen kuin S_2:n tapahtumista tiedetään mitään ja $H(S_1 | S_2)$ sen jälkeen. Erotus $H(S_1) - H(S_1 | S_2)$ mittaa jonkinlaista järjestelmien "yhteistä" informaatiota.

Täydellisten tapausjärjestelmien S_1 ja S_2 *keskinäisinformaatio* on

$$\boxed{H(S_1) - H(S_1 | S_2) = I(S_1, S_2).}$$

Lauseen **31** nojalla

$$H(S_1 | S_2) + H(S_2) = H(S_1 \cap S_2) = H(S_2 \cap S_1) = H(S_2 | S_1) + H(S_1)$$

eli

$$H(S_1) - H(S_1 \mid S_2) = H(S_2) - H(S_2 \mid S_1).$$

Näin ollen

$$\boxed{I(S_1,S_2) = I(S_2,S_1),}$$

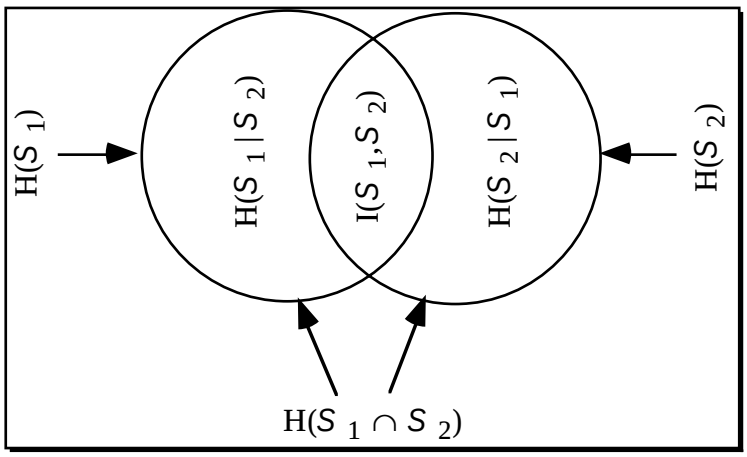
ts. tapausjärjestelmien keskinäisinformaatio ei riipu niiden järjestyksestä. Lauseen **31** nojalla edelleen

$$\boxed{I(S_1,S_2) = H(S_1) + H(S_2) - H(S_1 \cap S_2).}$$

Vielä Lauseesta **31** saadaan kaava

$$\boxed{I(S_1,S_2) = H(S_1 \cap S_2) - H(S_1 \mid S_2) - H(S_2 \mid S_1).}$$

Kaaviona:



LAUSE 32. $$\boxed{I(S_1,S_2) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log \frac{p_{ij}}{p_i q_j} = \operatorname{trace}(\mathbf{P} \log(\mathbf{Q}^{T} \lceil \mathbf{p} \rfloor^{-1}))}$$

(Jälleen sovitaan, että jos jokin todennäköisyyksistä p_{ij}, p_i tai q_j on =0, niin ko. termi summassa on =0. Huomaa, että jos $p_i = 0$ tai/ja $q_j = 0$, niin myös $p_{ij} = 0$.)

Todistus. Kokonaistodennäköisyyskaavan mukaisesti

$$\begin{aligned} I(S_1,S_2) &= H(S_1) + H(S_2) - H(S_1 \cap S_2) \\ &= -\sum_{i=1}^{n} p_i \log p_i - \sum_{j=1}^{m} q_j \log q_j + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij} \end{aligned}$$

$$= -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_i - \sum_{j=1}^{m}\sum_{i=1}^{n} p_{ij} \log q_j + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log p_{ij}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log \frac{p_{ij}}{p_i q_j}. \quad \blacksquare$$

Tapausjärjestelmien S_1 ja S_2 sanotaan olevan *riippumattomat,* jos

$$p_{ij} = p_i q_j \quad (i = 1,\ldots,n; j = 1,\ldots,m)$$

eli $\mathbf{P} = \mathbf{p}^T\mathbf{q}$, ts. jos järjestelmien tapaukset ovat parittain keskenään riippumattomat.

LAUSE 33. Jos S_1 ja S_2 ovat riippumattomat, niin $I(S_1,S_2) = 0$ (ja siis seurauksena $H(S_1 \mid S_2) = H(S_1)$ sekä $H(S_2 \mid S_1) = H(S_2)$ ja $H(S_1 \cap S_2) = H(S_1) + H(S_2)$).

Todistus. Seuraa suoraan Lauseesta **32**. $\blacksquare$

Yleisesti on voimassa

LAUSE 34. $\boxed{H(S_1 \mid S_2) \le H(S_1)}$ ja yhtäsuuruus pätee tarkalleen silloin, kun S_1 ja S_2 ovat riippumattomat. Sama asia toisin: $\boxed{I(S_1,S_2) \ge 0}$ ja yhtäsuuruus pätee tarkalleen silloin, kun S_1 ja S_2 ovat riippumattomat.

Todistus. Jos $p_{ij} = 0$, niin

$$-p_{ij} \ln \frac{p_{ij}}{p_i q_j} = 0 \le p_i q_j - 0 = p_i q_j - p_{ij}$$

ja yhtäsuuruus esiintyy tarkalleen silloin, kun $p_i q_j = 0 = p_{ij}$.

Jos taas $p_{ij} \ne 0$ (jolloin myös $p_i \ne 0$ ja $q_j \ne 0$), niin sivun 92 Apulauseen nojalla

$$-p_{ij} \ln \frac{p_{ij}}{p_i q_j} = p_{ij} \ln \frac{p_i q_j}{p_{ij}} \le p_{ij}\left(\frac{p_i q_j}{p_{ij}} - 1\right) = p_i q_j - p_{ij}$$

ja yhtäläisyys esiintyy tarkalleen silloin, kun $p_{ij} = p_i q_j$. Näin ollen nateissa

$$-I(S_1, S_2) = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(-p_{ij} \ln \frac{p_{ij}}{p_i q_j}\right) \le \sum_{i=1}^{n}\sum_{j=1}^{m}(p_i q_j - p_{ij})$$

$$= \sum_{i=1}^{n} p_i \sum_{j=1}^{m} q_j - \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} = 0$$

ja yhtäsuuruus pätee tarkalleen silloin, kun se pätee kaikille yhteenlaskettaville eli tarkalleen silloin, kun

$$p_{ij} = p_i q_j \quad (i = 1,\ldots,n; j = 1,\ldots,m)$$

eli tarkalleen silloin, kun S_1 ja S_2 ovat riippumattomat.

Lause pätee siis nateille ja näin ollen muillekin informaatioyksiköille. ■

Määritelmästä seuraa, että

$$\boxed{I(S_1,S_2) \leq \min(H(S_1),H(S_2)).}$$

5. Kolme tapausjärjestelmää: Data–Processing-lause

Tarkastellaan kolmea täydellistä tapausjärjestelmää

$$S_1 : A_1,\ldots,A_n ; \quad S_2 : B_1,\ldots,B_m \quad \text{ja} \quad S_3 : C_1,\ldots,C_r.$$

Yleisesti, jos $P(C) \neq 0$ ja $S : D_1,\ldots,D_s$ on täydellinen tapausjärjestelmä, niin

$$S\,|C : D_1|C ,\ldots, D_s|C$$

on myös täydellinen tapausjärjestelmä. Näin ollen $S_1|C_k$ ja $S_2|C_k$ ovat täydellisiä tapausjärjestelmiä, mikäli $P(C_k) \neq 0$ $(k = 1,\ldots,r)$.

Keskinäisinformaatio $I(S_1|C_k,S_2|C_k)$ mittaa S_1:n ja S_2:n keskimääräistä "yhteistä" informaatiota, kun tietyn S_3:n tapauksen C_k tiedetään sattuneen. Keskimäärin S_1:n ja S_2:n "yhteinen informaatio", kun tiedetään S_3:n tapauksen sattuneen, on

$$\boxed{\sum_{\substack{k=1 \\ P(C_k)\neq 0}}^{r} I(S_1|C_k, S_2|C_k)P(C_k) = I(S_1, S_2|S_3),}$$

ns. S_1:n ja S_2:n *ehdollinen keskinäisinformaatio* ehdolla S_3.

Jos $S_1|C_k$ ja $S_2|C_k$ ovat riippumattomat kaikille k:n arvoille $k = 1,\ldots,r$, joille $P(C_k) \neq 0$, ovat S_1 ja S_2 *riippumattomat ehdolla* S_3. Ts. S_1 ja S_2 ovat riippumattomat ehdolla S_3, jos

$$P(A_i \cap B_j \mid C_k) = P(A_i \mid C_k)P(B_j \mid C_k)$$

$$(i = 1,\ldots,n; j = 1,\ldots,m; k = 1,\ldots,r \text{ ja } P(C_k) \neq 0).$$

Jos $P(C_k) \neq 0$, niin Lauseen **34** nojalla $I(S_1,S_2 \mid C_k) \geq 0$ ja yhtäläisyys pätee tarkalleen silloin, kun $S_1 \mid C_k$ ja $S_2 \mid C_k$ ovat riippumattomat. Näin ollen saadaan välittömästi

LAUSE 35. $\boxed{I(S_1,S_2 \mid S_3) \geq 0}$ ja yhtäläisyys pätee tarkalleen silloin, kun S_1 ja S_2 ovat riippumattomat ehdolla S_3. ■

Lauseen **32** mukaan (olettaen, että $P(C_k) \neq 0$)

$$I(S_1|C_k, S_2|C_k) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(A_i \cap B_j|C_k)\log\frac{P(A_i \cap B_j|C_k)}{P(A_i|C_k)P(B_j|C_k)}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{P(A_i \cap B_j \cap C_k)}{P(C_k)}\log\frac{P(A_i \cap B_j|C_k)}{P(A_i|C_k)P(B_j|C_k)}$$

(missä yhteenlaskettava on =0, jos jokin esiintyvistä todennäköisyyksistä on nolla). Näin ollen

$$I(S_1, S_2|S_3) = \sum_{k=1}^{r}\sum_{i=1}^{n}\sum_{j=1}^{m} P(A_i \cap B_j \cap C_k)\log\frac{P(A_i \cap B_j|C_k)}{P(A_i|C_k)P(B_j|C_k)},$$

missä yhteenlaskettava on =0, jos jokin esiintyvistä todennäköisyyksistä on =0. (Huomaa, että jos $P(C_k) = 0$, niin myös $P(A_i \cap B_j \cap C_k) = 0$.)

LAUSE 36. $\boxed{I(S_1,S_2 \mid S_3) = I(S_1 \cap S_3,S_2) - I(S_2,S_3)}$

Todistus. Kokonaistodennäköisyyskaavan nojalla

$$I(S_1, S_2|S_3) = \sum_{k=1}^{r}\sum_{i=1}^{n}\sum_{j=1}^{m} P(A_i \cap B_j \cap C_k)\log\frac{\frac{P(A_i \cap B_j \cap C_k)}{P(C_k)}}{\frac{P(A_i \cap C_k)}{P(C_k)}\,\frac{P(B_j \cap C_k)}{P(C_k)}}$$

$$= \sum_{k=1}^{r}\sum_{i=1}^{n}\sum_{j=1}^{m} P(A_i \cap B_j \cap C_k)\left(\log\frac{P(A_i \cap B_j \cap C_k)}{P(A_i \cap C_k)P(B_j)} - \log\frac{P(B_j \cap C_k)}{P(B_j)P(C_k)}\right)$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{r}\sum_{j=1}^{m} P\big((A_i \cap C_k) \cap B_j\big)\log\frac{P\big((A_i \cap C_k) \cap B_j\big)}{P(A_i \cap C_k)P(B_j)}$$

$$-\sum_{j=1}^{m}\sum_{k=1}^{r}\sum_{i=1}^{n} P(A_i \cap B_j \cap C_k)\log\frac{P(B_j \cap C_k)}{P(B_j)P(C_k)}$$

$$= I(S_1 \cap S_3, S_2) - I(S_2, S_3). \quad \blacksquare$$

LAUSE 37. (DATA–PROCESSING–LAUSE) Jos S_1 ja S_2 ovat riippumattomat ehdolla S_3, niin

$$I(S_1,S_2) \leq I(S_2,S_3) \qquad \text{ja} \qquad I(S_1,S_2) \leq I(S_1,S_3).$$

Todistus. Jos S_1 ja S_2 ovat riippumattomat ehdolla S_3, niin (Lause **35**) $I(S_1, S_2 | S_3) = 0$. Lauseen **36** nojalla on tällöin

$$I(S_1 \cap S_3,S_2) = I(S_2,S_3).$$

Toisaalta (Lause **36**, vaihdetaan S_1 ja S_3 keskenään)

$$I(S_1 \cap S_3,S_2) = I(S_3 \cap S_1,S_2) = I(S_3,S_2 | S_1) + I(S_2,S_1) \geq I(S_1,S_2).$$

Siis $I(S_1,S_2) \leq I(S_2,S_3)$. Vaihtamalla S_1 ja S_2 keskenään saadaan

$$I(S_1, S_2) = I(S_2,S_1) \leq I(S_1,S_3). \quad \blacksquare$$

Lauseen nimi tulee seuraavasta tulkinnasta. Tapausjärjestelmä S_1 liittyy tietojen syöttöön, S_2 tulostukseen prosessoinnin jälkeen ja S_3 välitilaan prosessoinnin aikana. (Mieti miten riippumattomuusoletus liittyy tähän tilanteeseen.) Lause sanoo tällöin, että tietojen prosessointi ei voi lisätä informaatiota! Prosessoinnilla voi luonnollisesti muokata tietoja huomattavasti käyttökelpoisempaan muotoon.

Samalla todistuksella saadaan

LAUSE 38. (KONVEKSISUUSLAUSE) Jos S_1 ja S_2 ovat riippumattomat ehdolla S_3, niin

$$I(S_2,S_3) \geq I(S_2,S_3|S_1).$$

Todistus. Edellisen todistuksen nojalla

$$I(S_2,S_3) = I(S_3,S_2 | S_1) + I(S_2,S_1) \geq I(S_2,S_3 | S_1). \quad \blacksquare$$

Samantapaisia tuloksia voidaan todistaa useammallekin kuin kolmelle tapaussysteemille (ks. esimerkiksi MCELIECE).

6. Entropian yksikäsitteisyys

Informaatioteoria antaa useita perustavaa laatua olevia rajatuloksia tiedonsiirrossa (Data–Processing-lause, koodaus). Nämä menettäisivät merkityksensä, jos edellä oleva informaatiomitta ja entropia olisivat valittavissa oleellisesti toisella tavalla, joka olisi yhtä käyttökelpoinen, mutta johtaisi eri tuloksiin. Näin ei ole!

Ajatellaan täydellistä tapausjärjestelmää

$$S : A_1,\ldots,A_n \qquad (n \geq 2)$$

jonka todennäköisyysvektori on $\mathbf{p} = (p_1,\ldots,p_n)$. Jos $f(p_1,\ldots,p_n)$ on S :n entropiaa vastaava suure, niin se toteuttaa seuraavat hyvin luonnolliset vaatimukset:

(1) (jatkuvuus) $f(p_1,\ldots,p_n)$ on muuttujien $p_1,\ldots,p_n$ jatkuva funktio, kun $p_i \geq 0$ $(i = 1,\ldots,n)$ ja $p_1 + \cdots + p_n = 1$.

(2) (symmetrisyys) $f(p_1,\ldots,p_n)$ on symmetrinen funktio, ts. muuttujien $p_1,\ldots,p_n$ järjestyksen vaihtaminen ei muuta funktion arvoa.

(3) $f(p_1,\ldots,p_n)$ saa suurimman arvonsa, kun $p_1 = \cdots = p_n = 1/n$ ja mainittu suurin arvo on $\neq 0$.

(4) f ei muutu, jos täydelliseen tapausjärjestelmään lisätään mahdoton tapaus A_{n+1} (jolloin siis $p_{n+1} = 0$), ts.

$$f(p_1,\ldots,p_n,0) = f(p_1,\ldots,p_n).$$

Viidennen vaatimuksen esittämiseksi tarkastellaan kahta S :stä saatua täydellistä tapausjärjestelmää

$$S' : A_1,\ldots,A_{i-1},A_{i+1},\ldots,A_{j-1},C,A_{j+1},\ldots,A_n)$$

ja

$$S\,|\,C : A_1\,|\,C,\ldots,A_n\,|\,C,$$

missä $C = A_i \cup A_j$ ei ole mahdoton tapaus. Silloin

$$P(C) = P(A_i \cup A_j) = P(A_i) + P(A_j) = p_i + p_j \ (\neq 0).$$

Edelleen

$$P(A_i\,|\,C) = \frac{P\big(A_i \cap (A_i \cup A_j)\big)}{P(A_i \cup A_j)} = \frac{P(A_i)}{P(A_i \cup A_j)} = \frac{p_i}{p_i + p_j}$$

sekä vastaavasti

$$P(A_j \mid C) = \frac{p_j}{p_i + p_j}$$

ja

$$P(A_k \mid C) = \frac{P(A_k \cap (A_i \cup A_j))}{P(A_i \cup A_j)} = 0 \qquad (k = 1,\ldots,n; k \neq i,j).$$

S' saadaan samaistamalla A_i ja A_j yhdeksi tapaukseksi $A_i \cup A_j$. $S \mid C$ puolestaan saadaan olettamalla A_i:n tai A_j:n tapahtuneen. On luonnollista vaatia, että

$$S\text{:n entropia} = (S'\text{:n entropia}) + P(A_i \cup A_j) \times (S \mid C\text{:n entropia})$$

(muista, että entropia on keskimääräisen epävarmuuden mitta). Näin ollen vaaditaan, että

(5) (koherenssi)

$$f(p_1,\ldots,p_n) = f(p_1,\ldots,p_i - 1, p_i + 1,\ldots,p_j - 1, p_i + p_j, p_j + 1,\ldots,p_n) + (p_i + p_j) f\left(\frac{p_i}{p_i + p_j}, \frac{p_j}{p_i + p_j}\right),$$

kun $p_i + p_j \neq 0$.

LAUSE 39. (INFORMAATIOTEORIAN YKSIKÄSITTEISYYSLAUSE) Jos f toteuttaa eo. ehdot **(1)–(5)**, niin se on muotoa

$$f(p_1,\ldots,p_n) = -C \sum_{i=1}^{n} p_i \ln p_i,$$

missä C on positiivinen vakio.

Todistus. Merkitään

$$g(n) = f\left(\frac{1}{n},\ldots,\frac{1}{n}\right)$$

ja etsitään ensin g(n):n lauseke.

Vaihe 1 Näytetään, että $g(n) \leq g(n + 1)$:

$$g(n) = f\left(\frac{1}{n},\ldots,\frac{1}{n}\right) =_{(4)} f\left(\frac{1}{n},\ldots,\frac{1}{n},0\right) \leq_{(3)} f\left(\frac{1}{n+1},\ldots,\frac{1}{n+1}\right) = g(n + 1).$$

Vaihe 2 Etsitään g(n):n lauseke. Yleisesti, jos $p \neq 0$, niin

$$f(p_1,\ldots,p_{n-r},p,\ldots,p) =_{(5)} f(p_1,\ldots,p_{n-r},p,\ldots,p,2p) + (p+p)f\left(\frac{p}{p+p},\frac{p}{p+p}\right)$$

$$= f(p_1,\ldots,p_{n-r},p,\ldots,p,2p) + 2pf\left(\frac{1}{2},\frac{1}{2}\right)$$

$$=_{(5)} f(p_1,\ldots,p_{n-r},p,\ldots,p,3p) + (p+2p)f\left(\frac{p}{p+2p},\frac{2p}{p+2p}\right) + 2pf\left(\frac{1}{2},\frac{1}{2}\right)$$

$$= f(p_1,\ldots,p_{n-r},p,\ldots,p,3p) + 3pf\left(\frac{1}{3},\frac{2}{3}\right) + 2pf\left(\frac{1}{2},\frac{1}{2}\right).$$

Jatkamalla samalla tavoin saadaan lopulta kaava

A $$f(p_1,\ldots,p_{n-r},p,\ldots,p) = f(p_1,\ldots,p_{n-r},rp) + rpf\left(\frac{1}{r},\frac{r-1}{r}\right) + \cdots + 3pf\left(\frac{1}{3},\frac{2}{3}\right) + 2pf\left(\frac{1}{2},\frac{1}{2}\right).$$

Tätä kaavaa voidaan soveltaa g(n):ään valitsemalla $r = n - 1$:

B $$g(n) = f\left(\frac{1}{n},\ldots,\frac{1}{n}\right) = f\left(\frac{1}{n},\frac{n-1}{n}\right) + \frac{n-1}{n} f\left(\frac{1}{n-1},\frac{n-2}{n-1}\right) + \cdots + \frac{3}{n} f\left(\frac{1}{3},\frac{2}{3}\right) + \frac{2}{n} f\left(\frac{1}{2},\frac{1}{2}\right).$$

Kaavoista **A** ja **B** saadaan näin ollen kaava

C $$f(p_1,\ldots,p_{n-r},p,\ldots,p) = f(p_1,\ldots,p_{n-r},rp) + rpg(r)$$

(**B**:ssä n:n tilalle r).

Jos nyt $m \geq 2$ ja $r \geq 2$, niin

$$g(r^m) = f\left(\frac{1}{r^m},\ldots,\frac{1}{r^m}\right) =_{\mathbf{C}} f\left(\frac{1}{r^m},\ldots,\frac{1}{r^m},r\frac{1}{r^m}\right) + r\frac{1}{r^m} g(r)$$

$$=_{(2),\mathbf{C}} f\left(\frac{1}{r^m},\ldots,\frac{1}{r^m},r\frac{1}{r^m},\frac{1}{r^{m-1}}\right) + 2\frac{1}{r^{m-1}} g(r).$$

Jatkamalla samalla tavoin saadaan lopulta kaava

D $$g(r^m) = g(r^m - 1) + g(r)$$

(tähän tarvitaan $r^m - 1$ yo. operaation toistoa). Edelleen

$$g(r^m) =_{\mathbf{D}} g(r^m - 1) + g(r) =_{\mathbf{D}} g(r^m - 2) + 2g(r) =_{\mathbf{D}} \cdots =_{\mathbf{D}} g(r) + (m-1)g(r)$$

eli saadaan kaava

E $\quad g(r^m) = mg(r)$

(olettaen, että $m \geq 1$ ja $r \geq 2$).

Jos nyt $m \geq 1$ ja $r \geq 2$, niin voidaan valita sellainen luku $t \geq 1$, että

$$2^t \leq r^m \leq 2^{t+1}.$$

Silloin (Vaihe 1)

$$g(2^t) \leq g(r^m) \leq g(2^{t+1})$$

eli kaavan **E** nojalla

$$tg(2) \leq mg(r) \leq (t+1)g(2).$$

Tästä näkyy, että $g(2) \geq 0$, jolloin **(3)**:n nojalla itse asiassa $g(2) > 0$. Jaetaan nyt yo. kaksoisepäyhtälö puolittain $mg(2)$:llä:

$$\frac{t}{m} \leq \frac{g(r)}{g(2)} \leq \frac{t+1}{m}.$$

Toisaalta myös $t \ln 2 \leq m \ln r \leq (t+1)\ln 2$, joten samoin

$$\frac{t}{m} \leq \frac{\ln r}{\ln 2} \leq \frac{t+1}{m}.$$

Yhdistetään näin saadut kaksi kaksoisepäyhtälöä:

$$\frac{t}{m} - \frac{t+1}{m} \leq \frac{g(r)}{g(2)} - \frac{\ln r}{\ln 2} \leq \frac{t+1}{m} - \frac{t}{m}.$$

Annetaan tässä $m \to \infty$ (pitäen r "paikoillaan"):

$$\frac{g(r)}{g(2)} = \frac{\ln r}{\ln 2}$$

eli

$$g(r) = \frac{g(2)}{\ln 2} \ln r.$$

Valitaan nyt $C = g(2)/\ln 2 > 0$, jolloin

F $\quad g(r) = C \ln r \qquad (r = 2,3,\ldots).$

Vaihe 3 Tarkastellaan tapausta, jossa $p_1,\ldots,p_n$ ovat rationaalilukuja, ts. (tehdään samannimisiksi)

$$p_i = \frac{m_i}{M} \qquad (i = 1,\ldots,n).$$

Silloin $m_1 + \cdots + m_n = M$ ja **C**:n perusteella (valitaan $r = m_n, m_{n-1}, \ldots > 1$)

$$f(\tfrac{1}{M},\ldots,\tfrac{1}{M}) = f(\tfrac{1}{M},\ldots,\tfrac{1}{M},\tfrac{m_n}{M}) + \tfrac{m_n}{M}\, g(m_n)$$

$$= f(\tfrac{1}{M},\ldots,\tfrac{1}{M},\tfrac{m_{n-1}}{M},p_n) + \tfrac{m_{n-1}}{M}\, g(m_{n-1}) + p_n g(m_n)$$

$$= \cdots = f(p_1,\ldots,p_n) + p_1 g(m_1) + \cdots + p_n g(m_n).$$

(Jos m_i on 0 tai 1, jää vastaava askel pois, ts. sovitaan, että $0g(0) = 0$ ja $g(1) = 0$.) Siis

$$f(p_1,\ldots,p_n) = g(M) - p_1 g(m_1) - \cdots - p_n g(m_n)$$

$$=_{\mathbf{F}} C \ln M - p_1 C \ln m_1 - \cdots - p_n C \ln m_n$$

$$= C \ln M \sum_{i=1}^{n} p_i - C \sum_{i=1}^{n} p_i \ln m_i$$

$$= -C \sum_{i=1}^{n} p_i (\ln m_i - \ln M) = -C \sum_{i=1}^{n} p_i \ln p_i.$$

Lause siis pitää paikkansa rationaalisille todennäköisyyksille.

Vaihe 4 Siirrytään yleisiin todennäköisyyksiin. Mikä tahansa todennäköisyysvektori $(p_1,\ldots,p_n)$ saadaan rationaalisten todennäköisyysvektorien raja–arvona (miten?). Koska

$$-C \sum_{i=1}^{n} p_i \ln p_i$$

on jatkuva ja samoin $f(p_1,\ldots,p_n)$ (vaatimus **(1)**), seuraa tästä että yleisestikin

$$f(p_1,\ldots,p_n) = -C \sum_{i=1}^{n} p_i \ln p_i. \quad \blacksquare$$

Helposti voidaan tietysti todeta, että edellä määritelty entropia $H(S)$ toteuttaa vaatimukset **(1)–(5)**.

7. Jatkuvan satunnaismuuttujan entropia

Entropian yleistys täydelliseen tapausjärjestelmään, jossa on numeroituvasti ääretön määrä tapauksia

$$S : A_1, A_2, \ldots,$$

on suoraviivaista. Jos $p_1, p_2, \ldots$ ovat vastaavat todennäköisyydet, niin

$$H(S) = -\sum_{i=1}^{\infty} p_i \log p_i.$$

Huomaa, että tällainen entropia voi olla ääretön, toisin kuin äärelliselle tapausjärjestelmälle.

Jos merkitään satunnaismuuttujan x arvoksi i, mikäli A_i tapahtuu, niin sekä äärelliselle että äärettömälle tapausjärjestelmälle S

$$H(S) = E(-\log p_x).$$

Jatkuvalle satunnaismuuttujalle x, jonka tiheysfunktio on f(x), p_x:ää vastaa differentiaali f(x)dx ja $-\log(f(x)dx)$ ei ole määritelty! Jatkuvalle satunnaismuuttujalle ei olekaan samanlaista informaation ja entropian käsitettä kuin diskreeteille satunnaismuuttujille. Jos tavoitteena on vain entropioiden additiivinen vertailu (esimerkiksi maksimientropiamenetelmissä, ks. Luku **X**), voidaan kuitenkin menetellä seuraavasti. Jaetaan reaaliakseli Δ:n pituisiin väleihin $[i\Delta,(i+1)\Delta)$, $i = \ldots,-1,0,1,2,\ldots$, ja valitaan kultakin tällaiselta väliltä $[i\Delta,(i+1)\Delta)$ piste x_i, jolle

$$P\big(i\Delta \le x < (i+1)\Delta\big) = \int_{i\Delta}^{(i+1)\Delta} f(x)dx = f(x_i)\Delta = p_i$$

(integraalilaskennan väliarvolauseen nojalla tällainen x_i on olemassa). Otetaan tarkasteltavaksi täydellinen tapausjärjestelmä

$$S_\Delta : \ldots, A_{-1}, A_0, A_1, A_2, \ldots,$$

missä $A_i = \{i\Delta \le x < (i+1)\Delta\}$ $(i = \ldots,-1,0,1,2,\ldots)$. Silloin

$$H(S_\Delta) = -\sum_{i=-\infty}^{\infty} p_i \log p_i = -\sum_{i=-\infty}^{\infty} f(x_i)\Delta \log\big(f(x_i)\Delta\big)$$
$$= -\sum_{i=-\infty}^{\infty} f(x_i)\Delta \log f(x_i) - \log\Delta.$$

Vertailtaessa entropioita additiivisesti supistuu hankaluuksia aiheuttava termi $-\log \Delta$ pois ja rajalla $\Delta \to 0+$ saadaan vertailusuure, ns. x:n *differentiaalientropia*

$$\boxed{H(x) = -\int_{-\infty}^{\infty} f(x)\log f(x)dx = E\big(-\log f(x)\big).}$$

Vastaavasti saadaan jatkuvan satunnaisvektorin $\mathbf{x}$ differentiaalientropia

$$\boxed{H(\mathbf{x}) = -\int_{-\infty}^{\infty} f(\mathbf{x})\log f(\mathbf{x})d\mathbf{x} = E\big(-\log f(\mathbf{x})\big).}$$

Paitsi maksimientropiamenetelmissä, voi differentiaalientropiaa soveltaa tapauksissa, joissa on kyseessä entropioiden erotus. Tällaisia ovat esimerkiksi ehdollinen entropia sekä keskinäisinformaatio.

Huomaa, että differentiaalientropia voi olla ääretön tai negatiivinen, jopa $-\infty$. Differentiaalientropiaa on käsitelty laajemmin esimerkiksi viitteessä MCELIECE.

VIII LUKU

LÄHTEEN KOODAUS

1. Yleistä

Lähde (tarkemmin sanoen diskreetti lähde) on systeemi, joka tuottaa tietyn symbolijoukon eli aakkoston (*lähdeaakkoston*) $A = \{a_1,\ldots,a_N\}$ symboleja tietyllä todennäköisyydellä. Jos tämä todennäköisyys ei riipu lähteen aikaisemmin tuottamista symboleista, on kyseessä *muistiton* lähde, muuten *muistillinen.*

Lähteen tuottama symbolijono jaetaan n-pituisiin peräkkäisiin lohkoihin, ns. *lähdesanoihin,* merkitään

$$\mathbf{u} = u_1 \cdots u_n$$

(u_i on siis lähdesanan **u** i:s symboli). Vrt. Luvun **I** viestisanat ja viestiaakkosto.

Menetellään samoin kuin Luvussa **I** eli kun lohkon pituus n on valittu, syötetään lähteen tuottama symbolijono lähdesanoittain kooderiin. Kooderi muuntaa lähdesanan koodiaakkoston $C = \{c_1,\ldots,c_M\}$ sanaksi, ns. koodisanaksi. Koodisanat muodostavat ns. koodin. Jos koodin kaikki koodisanat ovat samanpituiset, on kyseessä lohkokoodi, muuten ns. *vaihtuvapituinen koodi.*

Tarkoitus luonnollisesti on, että koodattu viesti (lähdesana) voidaan saada esiin dekoodauksella joko aina tai ainakin tietyllä suurella todennäköisyydellä.

2. Muistiton lähde, lohkokoodi

Koska lähde on muistiton, on kunkin lähdesymbolin esiintymistodennäköisyys vakio. Siis tapaukset

$$A_i = \{\text{“lähdesymboli on } a_i\text{”}\} \quad (i = 1,\ldots,N)$$

muodostavat täydellisen tapausjärjestelmän A, jonka todennäköisyysvektori

$$\mathbf{p} = (p_1,\ldots,p_N)$$

tunnetaan. Jatkossa oletetaan, että kaikki lähdesymbolit ovat mahdollisia, ts.

$$p_i \neq 0 \qquad (i = 1,\ldots,N).$$

Lähteen entropia on tällöin $H(A)$. Jos lähdesanassa **u** lähdesymboli a_i esiintyy n_i kertaa (i = 1,...,N), niin

$$\begin{aligned}
P(\mathbf{u}) &= P(\text{“lähdesana on } \mathbf{u}\text{”}) \\
&= P(\text{“(lähdesanan) 1. symboli on } u_1\text{, 2. symboli on } u_2\text{,... ja n:s symboli on } u_n\text{”}) \\
&= P(\text{“n:s symboli on } u_n\text{”} \mid \text{“1. symboli on } u_1\text{,... ja n – 1:s symboli on } u_{n-1}\text{”}) \cdot \\
&\quad \cdot P(\text{“1. symboli on } u_1\text{,... ja n – 1:s symboli on } u_{n-1}\text{”}) \\
&= P(\text{“n:s symboli on } u_n\text{”}) P(\text{“1. symboli on } u_1\text{,... ja n – 1:s symboli on } u_{n-1}\text{”}) \\
&= \cdots = P(\text{“n:s symboli on } u_n\text{”}) P(\text{“n – 1:s symboli on } u_{n-1}\text{”}) \cdot \\
&\quad \cdots P(\text{“1. symboli on } u_1\text{”}) \\
&= p_1^{n_1} \cdots p_N^{n_N}.
\end{aligned}$$

Suurten lukujen lain mukaan

$$\frac{n_i}{n} \cong p_i \qquad (i = 1,\ldots,N),$$

kun n on suuri. (Valitaan suotuisaksi tapaukseksi symbolin a_i esiintyminen, jolloin $S_n = n_i$.) Siispä suurelle n:n arvolle on (todennäköisesti)

$$\begin{aligned}
P(\mathbf{u}) &= p_1^{n_1} \cdots p_N^{n_N} \cong p_1^{np_1} \cdots p_N^{np_N} = \left(p_1^{p_1} \cdots p_N^{p_N}\right)^n \\
&= 2^{\log_2\left(p_1^{p_1} \cdots p_N^{p_N}\right)^n} = 2^{n(-H(A))} = 2^{-nH(A)},
\end{aligned}$$

ts.

A $\quad -\dfrac{\log_2 P(\mathbf{u})}{n} \cong H(A)$ bittiä.

Tämä arvio tarvitaan kuitenkin tarkemmassa muodossa:

LAUSE 40. (OSITUSLAUSE) Kaikille positiivisille luvuille η ja π (miten tahansa pienille) on olemassa sellainen vakio $N_{\eta\pi}$, että

$$P\left(\left|\frac{\log_2 P(\mathbf{u})}{n} + H(A)\right| \geq \eta\right) < \pi,$$

kun $n > N_{\eta\pi}$. (Sama asia sanoin: Todennäköisyys, että arvio **A** ei päde tietyllä (miten tahansa pienellä) tarkkuudella saadaan miten tahansa lähelle nollaa kasvattamalla n:n arvoa.)

Todistus. Suurten lukujen lain mukaan

$$P\left(\left|\frac{n_i}{n} - p_i\right| < \varepsilon\right) > 1 - \delta$$

eli

$$P\left(\left|\frac{n_i}{n} - p_i\right| \geq \varepsilon\right) < \delta \qquad (i = 1,\ldots,N),$$

kun n on kyllin suuri ($n > N_{\varepsilon\delta}$).

Valitaan ensin ε niin pieneksi, että jos $\left|\frac{n_i}{n} - p_i\right| < \varepsilon$ $(i = 1,\ldots,N)$, niin

$$\left|\frac{\log_2 P(\mathbf{u})}{n} + H(A)\right| < \eta.$$

Tämä onnistuu vaikkapa valitsemalla

$$\varepsilon = \frac{\eta}{-\sum_{i=1}^{N} \log_2 p_i}.$$

Jos nimittäin $np_i - n\varepsilon < n_i < np_i + n\varepsilon$ $(i = 1,\ldots,N)$, niin silloin

$$\frac{1}{n}\log_2 P(\mathbf{u}) + H(A) = \frac{1}{n}\log_2\left(p_1^{n_1}\cdots p_N^{n_N}\right) + H(A)$$

$$= \frac{1}{n}\sum_{i=1}^{N} n_i \log_2 p_i + H(A) < \frac{1}{n}\sum_{i=1}^{N} (np_i - n\varepsilon)\log_2 p_i + H(A)$$

$$= \sum_{i=1}^{N} p_i \log_2 p_i - \varepsilon \sum_{i=1}^{N} \log_2 p_i + H(A) = \eta$$

ja vastaavasti

$$\frac{1}{n}\log_2 P(\mathbf{u}) + H(A) > \frac{1}{n}\sum_{i=1}^{N}(np_i + n\varepsilon)\log_2 p_i + H(A)$$

$$= \sum_{i=1}^{N} p_i \log_2 p_i + \varepsilon \sum_{i=1}^{N} \log_2 p_i + H(A) = -\eta.$$

Jos nyt tämän valinnan jälkeen

$$\left| \frac{\log_2 P(\mathbf{u})}{n} + H(A) \right| \geq \eta,$$

niin ainakin yhdelle i:n arvolle

$$\left| \frac{n_i}{n} - p_i \right| \geq \varepsilon.$$

Merkitään

$$P = \max_{i=1}^{N} P\left(\left| \frac{n_i}{n} - p_i \right| \geq \varepsilon \right).$$

Siispä (Suurten lukujen laki, valitaan $\delta = \pi/N$), $P < \pi/N$, kun n on kyllin suuri. Nyt

$$P\left(\left| \frac{\log_2 P(\mathbf{u})}{n} + H(A) \right| \geq \eta \right)$$

$$\leq P\left(\left| \frac{n_1}{n} - p_1 \right| \geq \varepsilon \text{ tai } \left| \frac{n_2}{n} - p_2 \right| \geq \varepsilon \text{ tai } \cdots \text{ tai } \left| \frac{n_N}{n} - p_N \right| \geq \varepsilon \right)$$

$$\leq \sum_{i=1}^{N} P\left(\left| \frac{n_i}{n} - p_i \right| \geq \varepsilon \right) \leq NP < \pi,$$

kun n on kyllin suuri ($n > N_{\varepsilon\delta} = N_{\eta\pi}$). (Summa-arvio löytyy tilastomatematiikan kurssilta.) ■

Palataan koodaukseen. Koodisanojen lukumäärä on M^m, jos koodin pituus (eli koodisanojen yhteinen pituus) on m. Lähdesanoja on vastaavasti N^n kpl. Jos nyt $M^m \geq N^n$ eli

$$\frac{m}{n}\log_2 M \geq \log_2 N,$$

niin jokaiselle lähdesanalle riittää oma koodisanansa ja asia on selvä.

Jos $M^m < N^n$, ei kaikille lähdesanoille riitä omaa koodisanaa. Tällöin suoritetaan <u>ositus</u>, ts. jaetaan lähdesanat kahteen luokkaan

$$R = \{\text{sanat, joilla on oma koodisana}\}$$

ja

$$W = \{\text{muut lähdesanat}\}.$$

Kaikille W:n sanoille ei ole omaa koodisanaa, ts. niille joudutaan antamaan jonkin toisen lähdesanan koodisana. Tämä johtaa <u>dekoodausvirheeseen</u>. R:n sanat taas dekoodataan (koodauksen jälkeen) oikein. Tavoitteena on luonnollisesti saada todennäköisyys $P_0 = P(\mathbf{u} \in W)$ (<u>virhetodennäköisyys</u>) pieneksi.

Kuten lukija arvannee, käytetään ositukseen Osituslausetta. Merkitään tätä varten

$$G = \left\{ \text{sanat } \mathbf{u}, \text{ joille } \left| \frac{\log_2 P(\mathbf{u})}{n} + H(A) \right| < \eta \right\}$$

ja

$$F = \{\text{muut lähdesanat}\}.$$

Silloin Osituslauseen nojalla (kun $n > N_{\eta\pi}$)

$$P(\mathbf{u} \in F) < \pi$$

ja jos $\mathbf{u} \in G$, niin

$$-\eta < \frac{\log_2 P(\mathbf{u})}{n} + H(A) < \eta$$

eli

$$-n\eta - nH(A) < \log_2 P(\mathbf{u}) < n\eta - nH(A)$$

eli

B $$2^{-n(\eta + H(A))} < P(\mathbf{u}) < 2^{n(\eta - H(A))}.$$

Merkitään nyt N_G:llä G:n sanojen lukumäärää ja

C $$\begin{cases} P_{max} = \max\limits_{\mathbf{u} \in G} P(\mathbf{u}) <_{\mathbf{B}} 2^{n(\eta - H(A))} \\ P_{min} = \min\limits_{\mathbf{u} \in G} P(\mathbf{u}) >_{\mathbf{B}} 2^{-n(\eta + H(A))}. \end{cases}$$

Silloin

$$1 \geq P(\mathbf{u} \in G) = \sum_{\mathbf{u} \in G} P(\mathbf{u}) \geq N_G P_{min} >_{\mathbf{C}} N_G 2^{-n(\eta + H(A))}$$

eli

D $N_G < 2^{n(\eta + H(A))}$.

LAUSE 41. (SHANNONIN LÄHTEEN KOODAUSLAUSE) Kaikille positiiviluvuille η ja π (miten tahansa pienille) on olemassa sellainen vakio $N_{\eta\pi}$, että

(i) jos $\frac{m}{n} \log_2 M \geq H(A) + \eta$, niin $P_0 < \pi$, ja

(ii) jos $\frac{m}{n} \log_2 M \leq H(A) - 2\eta$, niin $P_0 > 1 - 2\pi$ (olipa ositus mikä tahansa!),

kun $n > N_{\eta\pi}$.

Todistus. Valitaan $N_{\eta\pi}$ siten, että Osituslause toteutuu ja että $2^{-n\eta} \leq \pi$, kun $n > N_{\eta\pi}$.

(i) Jos $\frac{m}{n} \log_2 M \geq H(A) + \eta$, niin

$$M^m \geq 2^{n(H(A) + \eta)} >_{\mathbf{D}} N_G$$

ja valitaan $R = G$. Silloin

$$P_0 = P(\mathbf{u} \in F) < \pi.$$

(ii) Jos $\frac{m}{n} \log_2 M \leq H(A) - 2\eta$, niin

$$M^m \leq 2^{n(H(A) - 2\eta)}.$$

Niiden G:n sanojen lkm, joilla on oma koodisana, on silloin enintään M^m ja

$$P(\text{“}\mathbf{u} \in G \text{ ja } \mathbf{u}\text{:lla on oma koodisana”}) \leq M^m P_{max} \leq 2^{n(H(A) - 2\eta)} P_{max}$$

$$<_{\mathbf{C}} 2^{n(H(A) - 2\eta)} 2^{n(\eta - H(A))} = 2^{-n\eta} \leq \pi.$$

Edelleen (Kokonaistodennäköisyyskaavan nojalla)

$$\begin{aligned} 1 - P_0 &= P(\text{“}\mathbf{u}\text{:lla on oma koodisana”}) \\ &= P(\text{“}\mathbf{u} \in G \text{ ja } \mathbf{u}\text{:lla on oma koodisana”}) \\ &\quad + P(\text{“}\mathbf{u} \in F \text{ ja } \mathbf{u}\text{:lla on oma koodisana”}) \\ &< \pi + P(\text{“}\mathbf{u} \in F \text{ ja } \mathbf{u}\text{:lla on oma koodisana”}) \end{aligned}$$

$$\leq \pi + P(\mathbf{u} \in F) < 2\pi$$

eli $P_0 > 1 - 2\pi$. ■

Jos nyt $\frac{m}{n} \log_2 M > H(A)$, niin löytyy sellainen η, että

$$\frac{m}{n} \log_2 M > H(A) + \eta.$$

Tällöin myös

$$\frac{km}{kn} \log_2 M > H(A) + \eta$$

ja Lähteen koodauslause saadaan käyttöön kasvattamalla k:ta, ts. pidentämällä lähde- ja koodisanoja samassa suhteessa. Virhetodennäköisyys saadaan tällöin miten tahansa pieneksi (k:ta kasvattamalla). Lauseen kohta **(ii)** osoittaa, ettei tämä onnistu, jos $\frac{m}{n} \log_2 M < H(A)$, itse asiassa päinvastoin. (Tapaus $\frac{m}{n} \log_2 M = H(A)$ ei juuri esiinny, sillä $H(A)/\log_2 M$ on enimmäkseen irrationaaliluku.)

3. Muistiton lähde, vaihtuvapituinen koodi

Vaihtuvapituisen koodin sanat voivat olla eripituisia. Mikä tahansa äärellinen koodisymboleista muodostettujen sanojen joukko ei kuitenkaan ole koodi: *Koodin* vaatimuksena on, että koodattu viesti on yksikäsitteisesti dekoodattavissa, ts. että mikä tahansa koodisymbolien jono on enintään yhdellä tavalla tulkittavissa koodisanojen jonoksi.

Jos $\mathbf{w} = w_1 \cdots w_\ell$ on koodisana (jolloin siis $w_1, \ldots, w_\ell$ ovat koodisymboleja), niin sanat

$$\mathbf{w},\ w_1 \cdots w_{\ell-1},\ w_1 \cdots w_{\ell-2}, \ldots,\ w_1 w_2,\ w_1$$

(ℓ kpl) ovat ns. $\mathbf{w}$:n *prefiksit*. Äärellinen sanojen joukko W on ns. *prefiksikoodi*, jos mikään W:n sana ei ole toisen W:n sanan prefiksi.

LAUSE 42. Prefiksikoodi on koodi.

Todistus. Pitää näyttää, että prefiksikoodin sanoista yhdistämällä ei saada kahdella eri tavalla yhdistäen samaa sanaa: Mikäli $\mathbf{w}_{i_1}, \ldots, \mathbf{w}_{i_p}$ ovat prefiksikoodin W sanoja, niin sanaa

$$\mathbf{w} = \mathbf{w}_{i_1} \mathbf{w}_{i_2} \cdots \mathbf{w}_{i_p}$$

ei voida kirjoittaa muulla tavoin W:n sanojen yhdyssanaksi. Jos nimittäin

$$\mathbf{w} = \mathbf{w}_{j_1}\mathbf{w}_{j_2}\cdots\mathbf{w}_{j_q},$$

missä myös $\mathbf{w}_{j_1},\ldots,\mathbf{w}_{j_q} \in W$, niin sanoista $\mathbf{w}_{i_1}$ ja $\mathbf{w}_{j_1}$ toinen on toisen prefiksi ja siis (prefiksikoodi) $\mathbf{w}_{i_1} = \mathbf{w}_{j_1}$. Näin ollen

$$\mathbf{w}_{i_2}\cdots\mathbf{w}_{i_p} = \mathbf{w}_{j_2}\cdots\mathbf{w}_{j_q}.$$

Mutta samalla perusteella $\mathbf{w}_{i_2} = \mathbf{w}_{i_2}$ ja jne.. Siispä q = p ja

$$\mathbf{w}_{i_1} = \mathbf{w}_{j_1},\ldots,\mathbf{w}_{i_p} = \mathbf{w}_{j_p}. \quad \blacksquare$$

Kaikki koodit eivät ole prefiksikoodeja.

LAUSE 43. (KRAFTIN LAUSE) Prefiksikoodi, jonka koodisanojen pituudet ovat $\ell_1,\ldots,\ell_k$, on olemassa tarkalleen silloin, kun

$$\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}} \le 1 \qquad \text{(ns. } \textit{Kraftin ehto}\text{)}.$$

(M on tässäkin koodisymbolien lkm.)

Todistus. Vaihtamalla tarvittaessa koodisanojen järjestystä voidaan olettaa, että

$$\ell_1 \le \ell_2 \le \cdots \le \ell_k.$$

Oletetaan ensiksi, että Kraftin ehto toteutuu, ts. (kun kerrotaan puolittain M^{ℓ_k}:lla)

$$M^{\ell_k-\ell_1} + M^{\ell_k-\ell_2} + \cdots + M^{\ell_k-\ell_{k-1}} + 1 \le M^{\ell_k}.$$

ℓ_k:n pituisia sanoja on M^{ℓ_k} kpl, merkitään niiden joukkoa V:llä. Jos nyt valitaan joukko sanoja W sillä tavalla, että kullakin V:n sanalla on enintään yksi W:n sana prefiksinä ja W:n sanojen pituus ei ylitä ℓ_k:ta, niin W on prefiksikoodi. Näytetään, että tämä voidaan tehdä siten, että W:n sanojen pituudet ovat $\ell_1,\ldots,\ell_k$:

1) Merkitään $V_1 = V$. Valitaan V_1:stä sana ja sen ℓ_1:n pituinen prefiksi $\mathbf{w}_1$. Poistetaan V_1:stä kaikki sanat, joiden prefiksi $\mathbf{w}_1$ on. Näitä on $M^{\ell_k-\ell_1}$ kpl. Jäljelle jäävän joukon V_2 sanojen lkm on $M^{\ell_k} - M^{\ell_k-\ell_1}$.

2) Valitaan V_2:sta sana ja sen ℓ_2:n pituinen prefiksi $\mathbf{w}_2$. (Kraftin ehdon nojalla tämä on mahdollista.) Poistetaan V_2:sta sanat, joiden prefiksi $\mathbf{w}_2$ on. Näitä on $M^{\ell_k-\ell_2}$ kpl.

3) Jne.. Lopulta saadaan haluttu prefiksikoodi $W = \{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$.

Oletetaan toiseksi, että $W = \{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ on prefiksikoodi, jonka sanojen pituudet ovat $\ell_1,\ldots,\ell_k$. Kullakin V:n sanalla on enintään yksi W:n sana prefiksinä. Sanoja, joiden prefiksinä on $\mathbf{w}_i$, on $M^{\ell_k-\ell_i}$ kpl ($i = 1,\ldots,k$). Siis

$$M^{\ell_k-\ell_1} + M^{\ell_k-\ell_2} + \cdots + M^{\ell_k-\ell_{k-1}} + 1 \le M^{\ell_k}.$$

eli Kraftin ehto toteutuu. (Huomaa, että ainoa V:n sana, jonka prefiksi on $\mathbf{w}_k$, on $\mathbf{w}_k$ itse.) ■

Luonnollisesti myös sanajoukot, jotka eivät ole koodeja, voivat toteuttaa Kraftin ehdon. Toisaalta jokainen koodi toteuttaa Kraftin ehdon (!), kuten seuraavassa lauseessa osoitetaan.

LAUSE 44. (MCMILLANIN LAUSE) Jokainen koodi toteuttaa Kraftin ehdon.

Todistus. Tarkastellaan koodia $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$, jonka koodisanojen pituudet ovat $\ell_1 \le \ell_2 \le \cdots \le \ell_k$. Otetaan mielivaltainen positiivinen kokonaisluku r (lopulta $r \to \infty$). Silloin

$$\left(\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}}\right)^r = \underbrace{\left(\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}}\right)\cdots\left(\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}}\right)}_{r \text{ kpl}}$$

$$= \sum_{i_1=1}^{k}\sum_{i_2=1}^{k}\cdots\sum_{i_r=1}^{k}\frac{1}{M^{\ell_{i_1}}}\frac{1}{M^{\ell_{i_2}}}\cdots\frac{1}{M^{\ell_{i_r}}} = \sum_{i_1=1}^{k}\sum_{i_2=1}^{k}\cdots\sum_{i_r=1}^{k} M^{-(\ell_{i_1}+\ell_{i_2}+\cdots+\ell_{i_r})}.$$

Summa $\ell_{i_1} + \ell_{i_2} + \cdots + \ell_{i_r}$ on r:n koodisanan yhdyssanan $\mathbf{w}_{i_1}\mathbf{w}_{i_2}\cdots\mathbf{w}_{i_r}$ pituus. Indeksien $i_1,\ldots,i_r$ kulkiessa toisistaan riippumatta arvot $1,\ldots,k$ saadaan kaikki r:n koodisanan yhdyssanat (mahdolliset toistot mukaanlukien).

Merkitään nyt

s_j = niiden r:stä koodisanasta eri tavoin yhdistämällä saatujen sanojen lkm (mahdolliset toistot mukaanlukien), joiden pituus on j.

Ilmeisesti mahdolliset j:n arvot ovat $j = r\ell_1,\ldots,r\ell_k$. Näin ollen

$$\left(\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}}\right)^r = \sum_{j=r\ell_1}^{r\ell_k} s_j M^{-j}.$$

Mutta koodi on yksikäsitteisesti dekoodattavissa, joten kahdella eri tavalla r:stä koodisanasta yhdistämällä ei voi saada samaa sanaa (ts. toistoja ei itse asiassa ole). Siis

$$s_j \le \text{j:n pituisten sanojen lkm} = M^j \qquad (j = r\ell_1,\ldots,r\ell_k)$$

ja

$$\left(\frac{1}{M^{\ell_1}}+\frac{1}{M^{\ell_2}}+\cdots+\frac{1}{M^{\ell_k}}\right)^r \le \sum_{j=r\ell_1}^{r\ell_k} M^j M^{-j} = r\ell_k - r\ell_1 + 1 \le r\ell_k .$$

Otetaan puolittain r:s juuri ja annetaan $r \to \infty$:

$$\frac{1}{M^{\ell_1}}+\frac{1}{M^{\ell_2}}+\cdots+\frac{1}{M^{\ell_k}} \le \sqrt[r]{r\ell_k} \to 1,$$

sillä (l'Hospitalin sääntö)

$$\lim_{r\to\infty} \ln \sqrt[r]{r\ell_k} = \lim_{r\to\infty} \frac{\ln r - \ln \ell_k}{r} = \lim_{r\to\infty} \frac{\frac{1}{r}-0}{1} = 0 .$$

Koska arvion vasen puoli ei riipu r:stä, pätee arvio sille myös rajalla $r \to \infty$, ts. Kraftin ehto on voimassa. ■

LAUSE 45. Jokainen koodi voidaan sananpituudet säilyttäen korvata prefiksikoodilla.

Todistus. Kraftin lause ja McMillanin lause. ■

Näin ollen voidaan tämä luvun tavoitteita ajatellen rajoittua prefiksikoodeihin.

4. Optimikoodaus

Kutakin koodisanaa $\mathbf{w}_i$ (i = 1,...,k) vastaa kääntäen yksikäsitteisesti lähdesana $\mathbf{u}_i$. Merkitään

$$P_i = P(\text{“lähdesana on } \mathbf{u}_i\text{”}) = P(\text{“koodisana on } \mathbf{w}_i\text{”}) \qquad (i = 1,\ldots,k),$$

$U = \{\mathbf{u}_1,\ldots,\mathbf{u}_k\}$ (lähdesanojen joukko) ja

$$H(U) = -\sum_{i=1}^{k} P_i \log P_i$$

(*lähdesanaentropia*). H(U):ta ei pidä sekoittaa lähteen entropiaan H(A). Käyttäen toistuvasti sitä tietoa (Lause **33**), että jos tapausjärjestelmät S_1 ja S_2 ovat riippumattomat, niin $H(S_1 \cap S_2) = H(S_1) + H(S_2)$, nähdään, että itse asiassa $H(U) = nH(A)$ (missä n on lähdesanan pituus). Merkitään edelleen

$$\bar{\ell} = P_1\ell_1 + \cdots + P_k\ell_k$$

(*koodisanojen keskipituus*), missä $\ell_1,\ldots,\ell_k$ ovat koodisanojen pituudet. Jatkossa oletetaan, että $P_1,\ldots,P_k \neq 0$, ts. että kaikki lähdesanat ovat mahdollisia.

Optimikoodaus tarkoittaa, että koodi valitaan siten, että $\bar{\ell}$ on pienin mahdollinen. Lauseen **45** nojalla voidaan rajoittua prefiksikoodeihin. Tehtävänä on näin valita sellaiset kokonaisluvut $\ell_1,\ldots,\ell_k$, että

$$\begin{cases} \bar{\ell} = P_1\ell_1 + \cdots + P_k\ell_k = \min! \\ \dfrac{1}{M^{\ell_1}} + \dfrac{1}{M^{\ell_2}} + \cdots + \dfrac{1}{M^{\ell_k}} \leq 1 \\ \ell_1,\ldots,\ell_k \geq 1 \end{cases}$$

Optimikoodaus siis tiivistää viestin mahdollisimman tehokkaasti (keskimäärin).

$\bar{\ell}$ voidaan arvioida H(U):n avulla alhaalta olipa koodi mikä tahansa (optimaalinen tai ei). Käytetään ensin natteja:

$$H(U) - \bar{\ell}\ \ln M = -\sum_{i=1}^{k} P_i \ln P_i - \left(\sum_{i=1}^{k} P_i \ell_i\right) \ln M$$

$$= -\sum_{i=1}^{k} P_i (\ln P_i + \ell_i \ln M) = \sum_{i=1}^{k} P_i \ln \frac{1}{P_i M^{\ell_i}}$$

$$\leq \sum_{i=1}^{k} P_i \left(\frac{1}{P_i M^{\ell_i}} - 1\right) = \sum_{i=1}^{k} \frac{1}{M^{\ell_i}} - \sum_{i=1}^{k} P_i \leq 0.$$

Siis nateissa $\bar{\ell} \ln M \geq H(U)$ ja näin ollen mielivaltaiselle kantaluvulle

$$\boxed{\bar{\ell} \geq \frac{H(U)}{\log M}\cdot}$$

Todettakoon vielä, että arviossa esiintyy yhtäsuuruus tarkalleen silloin, kun

$$\frac{1}{P_i M^{\ell_i}} = 1 \quad (i = 1,\ldots,k) \quad \text{ja} \quad \sum_{i=1}^{k} \frac{1}{M^{\ell_i}} = 1$$

eli kun $\boxed{P_i = 1 / M^{\ell_i}}$ $(i = 1,\ldots,k)$. Jos todennäköisyydet $P_1,\ldots,P_k$ ovat "sopimattomia" (eivät M:n negatiivisia potensseja), niin tähän minimiarvoon ei voida päästä. Toisaalta kuitenkin optimikoodauksessa päästään melko lähelle:

LAUSE 46. (SHANNONIN KOODAUSLAUSE) Optimikoodille

$$\boxed{\frac{H(U)}{\log M} \le \bar{\ell} < \frac{H(U)}{\log M} + 1}$$

ja näin ollen

$$\boxed{\lim_{n \to \infty} \frac{\bar{\ell}}{n} = \frac{H(A)}{\log M}.}$$

Todistus. Alaraja tuli jo todettua yllä. Ylärajan todistamiseksi valitaan kullekin todennäköisyydelle P_i lähin M:n negatiivinen potenssi, ts. kokonaisluku $\ell_i \ge 1$ siten, että

$$\frac{1}{M^{\ell_i}} \le P_i < \frac{1}{M^{\ell_i - 1}} \quad \text{eli} \quad M^{\ell_i} < \frac{M}{P_i} \le M^{\ell_i + 1}$$

$(i = 1,\ldots,k)$. Silloin

$$\frac{1}{M^{\ell_1}} + \frac{1}{M^{\ell_2}} + \cdots + \frac{1}{M^{\ell_k}} \le P_1 + \cdots + P_k = 1,$$

joten Kraftin ehto toteutuu ja (Kraftin lause) on olemassa prefiksikoodi, jonka koodisanojen pituudet ovat $\ell_1,\ldots,\ell_k$. Asetetaan ko. prefiksikoodin sanat $\mathbf{w}_1,\ldots,\mathbf{w}_k$ vastaamaan lähdesanoja siten, että

$$P(\text{"koodisana on } \mathbf{w}_i\text{"}) = P_i \qquad (i = 1,\ldots,k).$$

Silloin optimikoodin koodisanojen keskipituus on

$$\le \sum_{i=1}^{k} P_i \ell_i = \frac{1}{\log M} \sum_{i=1}^{k} P_i \ell_i \log M = \frac{1}{\log M} \sum_{i=1}^{k} P_i \log M^{\ell_i}$$

$$< \frac{1}{\log M} \sum_{i=1}^{k} P_i \log \frac{M}{P_i} = \frac{1}{\log M} \sum_{i=1}^{k} P_i (\log M - \log P_i)$$

$$= \sum_{i=1}^{k} P_i - \frac{1}{\log M} \sum_{i=1}^{k} P_i \log P_i = 1 + \frac{H(U)}{\log M} . \quad \blacksquare$$

5. Huffman–koodaus

Järjestetään lähdesanojen todennäköisyydet vähenevästi (tarvittaessa indeksointia muuttamalla):

$$P_1 \geq P_2 \geq \ldots \geq P_k \ (\geq 0).$$

Vastaavat koodisanat ovat $\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_k$ ja näiden pituudet $\ell_1,\ldots,\ell_k$.

APULAUSE. Koodi $\{\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_k\}$ voidaan valita siten, että se on optimaalinen prefiksikoodi, missä

(i) $\ell_1 \leq \cdots \leq \ell_{k-s} \leq \ell_{k-s+1} = \cdots = \ell_k$, missä s valitaan siten, että $2 \leq s \leq M$ ja $s \equiv k \pmod{M-1}$, ja

(ii) $\mathbf{w}_{k-s+1},\ldots,\mathbf{w}_k$ eroavat toisistaan vain viimeisessä symbolissa: $\mathbf{w}_{k-s+i}$:ssä se on c_i $(i = 1,\ldots,s)$. Sanojen $\mathbf{w}_{k-s+1},\ldots,\mathbf{w}_k$ yhteinen $\ell_k - 1$-pituinen prefiksi ei ole minkään sanan $\mathbf{w}_1,\ldots,\mathbf{w}_{k-s}$ prefiksi.

Todistus. Lauseen **45** nojalla on olemassa optimaalinen prefiksikoodi, sanotaan

$$\{\mathbf{w}'_1,\ldots,\mathbf{w}'_k\},$$

jonka sanojen pituudet ovat $\ell'_1,\ldots,\ell'_k$. Jos nyt jollekin i:lle $\ell'_i > \ell'_{i+1}$, niin vaihdetaan $\mathbf{w}'_i$ ja $\mathbf{w}'_{i+1}$, ts. lähdesana $\mathbf{u}_i$ koodataan $\mathbf{w}'_{i+1}$:ksi ja lähdesana $\mathbf{u}_{i+1}$ koodataan $\mathbf{w}'_i$:ksi. Koodisanojen keskipituuden muutos on

$$P_i\ell'_{i+1} + P_{i+1}\ell'_i - (P_i\ell'_i + P_{i+1}\ell'_{i+1}) = (P_i - P_{i+1})(\ell'_{i+1} - \ell'_i) \leq 0.$$

Toisaalta koodi oli optimaalinen, joten ko. muutos on =0 ja koodi pysyy vaihdonkin jälkeen optimaalisena (prefiksi)koodina. Toistamalla vaihto–operaatio (tarvittaessa) kyllin monta kertaa voidaan olettaa, että

$$\ell_1 \leq \cdots \leq \ell_k .$$

Näinkin saatuja optimikoodeja voi hyvinkin olla useita. Valitaankin jatkossa käytetty optimikoodi siten, että $\ell_1 + \cdots + \ell_k$ on pienin mahdollinen, ja merkitään

(*) $$\Delta = M^{\ell_k} - \sum_{i=1}^{k} M^{\ell_k - \ell_i} .$$

McMillanin lauseen nojalla $\Delta \geq 0$. Jos toisaalta olisi $\Delta \geq M - 1$, niin pituudet $\ell_1,\ldots,\ell_{k-1},\ell_k - 1$ toteuttaisivat Kraftin ehdon, mikä on mahdotonta, koska $\ell_1+\cdots+\ell_k$ oli pienin mahdollinen. Siispä $2 \leq M - \Delta \leq M$.

Merkitään nyt r:llä pituutta ℓ_k olevien koodisanojen lukumäärää. Silloin $r \geq 2$. Muussa tapauksessa voitaisiin $\mathbf{w}_k$:n viimeinen symboli poistaa ja saada prefiksikoodi (mahdotonta, koska $\ell_1+\cdots+\ell_k$ oli pienin mahdollinen). Yhtälöstä **(*)** seuraa, että $\Delta \equiv -r \pmod M$ eli $r \equiv M - \Delta \pmod M$. Koska toisaalta $2 \leq M - \Delta \leq M$, on r kirjoitettavissa muotoon

$$r = tM + (M - \Delta),$$

missä $t \geq 0$. Yhtälöstä **(*)** seuraa edelleen, että $\Delta \equiv 1 - k \pmod{M-1}$ (sillä $M \equiv 1 \pmod{M-1}$) eli $M - \Delta \equiv k \pmod{M-1}$. Näin nähdään, että $M - \Delta$ on juuri **(i)**:ssä mainittu luku s ja $r \geq s$.

Tarvittaessa indeksointia vaihtamalla voidaan olettaa, että koodin ℓ_k-pituisista sanoista $\mathbf{w}_{k-r+1},\ldots,\mathbf{w}_k$ sanat $\mathbf{w}_{k-t},\ldots,\mathbf{w}_k$ ovat sellaisia, että niiden $\ell_k - 1$-pituiset prefiksit $\mathbf{z}_1,\ldots,\mathbf{z}_{t+1}$ ovat keskenään erilaiset. Muista, että $r = tM + s$. Korvataan nyt sanat $\mathbf{w}_{k-r+1},\ldots,\mathbf{w}_k$ sanoilla $\mathbf{z}_1c_1,\ldots,\mathbf{z}_1c_M$, $\mathbf{z}_2c_1,\ldots,\mathbf{z}_2c_M,\ldots,\mathbf{z}_tc_1,\ldots,\mathbf{z}_tc_M,\mathbf{z}_{t+1}c_1,\ldots,\mathbf{z}_{t+1}c_s$. Tämä ei hävitä koodin prefiksisyyttä eikä optimaalisuutta. Saatu koodi on vaadittu $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$. ■

Tarkastellaan sanoja $\mathbf{v}_1,\ldots,\mathbf{v}_{k-s+1}$, jotka saadaan Apulauseen koodista $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ seuraavasti:

(a) $\mathbf{v}_1 = \mathbf{w}_1,\ldots,\mathbf{v}_{k-s} = \mathbf{w}_{k-s}$ ja

(b) $\mathbf{v}_{k-s+1}$ saadaan, kun poistetaan $\mathbf{w}_k$:sta sen viimeinen symboli $(= c_s)$.

$\{\mathbf{v}_1,\ldots,\mathbf{v}_{k-s+1}\}$ on prefiksikoodi. Ajatellaan sillä koodattavaksi $k - s + 1$ lähdesanaa, joiden todennäköisyydet ovat

$$P_1,\ldots,P_{k-s},P_{k-s+1}+\cdots+P_k.$$

Koodisanojen keskipituus on tällöin

$$\bar{\ell}' = \sum_{i=1}^{k-s} P_i\ell_i + \sum_{i=k-s+1}^{k} P_i(\ell_k - 1) = \bar{\ell} - \sum_{i=k-s+1}^{k} P_i.$$

Koodi $\{\mathbf{v}_1,\ldots,\mathbf{v}_{k-s+1}\}$ on näin ollen myös optimaalinen. (Muutoin olisi jokin pienemmän koodisanojen keskipituuden omaava prefiksikoodi $\{\mathbf{v}'_1,\ldots,\mathbf{v}'_{k-s+1}\}$ ja prefiksikoodin $\{\mathbf{v}'_1,\ldots,\mathbf{v}'_{k-s},\mathbf{v}'_{k-s+1}c_1,\ldots,\mathbf{v}'_{k-s+1}c_s\}$ koodisanojen keskipituus olisi pienempi kuin $\bar{\ell}' + P_{k-s+1}+\cdots+P_k = \bar{\ell}$.)

Toisaalta, jos $\{\mathbf{v}'_1,\ldots,\mathbf{v}'_{k-s+1}\}$ on optimaalinen prefiksikoodi (vastaten lähdesanatodennäköisyyksiä $P_1,\ldots,P_{k-s},P_{k-s+1}+\cdots+P_k$), niin sen koodisa-

nojen keskipituus on $\overline{\ell}'$ ja $\{\mathbf{v}'_1,\ldots,\mathbf{v}'_{k-s},\mathbf{v}'_{k-s+1}c_1,\ldots,\mathbf{v}'_{k-s+1}c_s\}$ on optimaalinen prefiksikoodi (vastaten lähdesanatodennäköisyyksiä $P_1,\ldots,P_k$). Muussa tapauksessa olisi Apulauseen antama optimikoodi, jonka koodisanojen keskipituus olisi pienempi kuin $\overline{\ell}' + P_{k-s+1} + \cdots + P_k$ ja tästä saataisiin kohtien **(a)** ja **(b)** avulla vielä koodia $\{\mathbf{v}'_1,\ldots,\mathbf{v}'_{k-s+1}\}$ "optimaalisempi" koodi.

Huffman–koodaus on seuraava rekursio, joka yllä esitetyn mukaan tuottaa optimikoodin saatuaan syöttönä lähdesanatodennäköisyydet $P_1,\ldots,P_k$ (vähenevässä järjestyksessä):

(1) Jos $k \le M$, niin valitaan $\mathbf{w}_1 = c_1,\ldots,\mathbf{w}_k = c_k$ ja lopetetaan.

(2) Muutoin koodi $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$ (vastaten lähdesanatodennäköisyyksiä $P_1,\ldots,P_k$) saadaan, kun koodi $\{\mathbf{v}_1,\ldots,\mathbf{v}_{k-s+1}\}$ tunnetaan:

$$\mathbf{w}_1 = \mathbf{v}_1 ,\ldots, \mathbf{w}_{k-s} = \mathbf{v}_{k-s} ,$$

$$\mathbf{w}_{k-s+1} = \mathbf{v}_{k-s+1}c_1 ,\ldots, \mathbf{w}_k = \mathbf{v}_{k-s+1}c_s.$$

Koodin $\{\mathbf{v}_1,\ldots,\mathbf{v}_{k-s+1}\}$ saamiseksi lasketaan s sekä asetetaan

$$Q_1 \leftarrow P_1 ,\ldots, Q_{k-s} \leftarrow P_{k-s} ,$$

$$Q_{k-s+1} \leftarrow P_{k-s+1} + \cdots + P_k.$$

Edelleen asetetaan todennäköisyyksien $P_1,\ldots,P_{k-s+1}$ arvoiksi $Q_1,\ldots,Q_{k-s+1}$ vähenevässä järjestyksessä, $k \leftarrow k-s+1$ ja palataan kohtaan **(1)**.

HUOM! Ensiksi valittava s toteuttaa ehdon $s \equiv k \pmod{M-1}$, jolloin "uusi k" eli $k-s+1$ on $\equiv 1 \pmod{M-1}$. Näin ollen seuraava s:n arvo onkin M. Itse asiassa näin jatkaen todetaan, että kaikki muutkin uudet s:n arvot ovat =M ja siis vain ensimmäinen s:n arvo pitää laskea! Huomaa myös, että tapauksessa M = 2 (binäärinen Huffman-koodi) aina s = 2.

6. Muita koodausmenettelyjä

Juoksulukukoodaus

Juoksulukukoodauksessa ajatellaan lähdesanat vaihtuvapituisiksi ja koodisanat ovat vakiopituisia. Koodausta käytetään silloin, kun jokin lähdesymboleista esiintyy huomattavasti muita useammin. Tarkastellaan esimerkkinä binäärisen lähteen tapausta, missä 1 esiintyy huomattavasti useammin kuin 0. Valitaan juoksun maksimipituus n. Koska

$$J = \{0,10,1^20,1^30,\ldots,1^{n-1}0,1^n\}$$

on prefiksikoodi, voidaan lähteen tuottama bittijono yksikäsitteisesti jakaa J :n sanoiksi. Asetetaan sitten J :n sanaa 1^i0 tai 1^i vastaamaan *juoksuluvun* i binääriesitys täydennettynä alkunollilla maksimipituuteen $m = \lfloor \log_2 n \rfloor + 1$. Tämä koodaus ei yleensä yllä lähellekään Huffman-koodauksen "pakkauskykyä", mutta on nopeampi suorittaa.

Aritmeettinen koodaus

Muuttamalla koodausmenettely Huffman-koodauksesta poikkeavaksi voidaan lähestyä Shannonin koodauslauseen alarajaa $H(U)/\log M$. Eräs tällainen menettely on *aritmeettinen koodaus*. Lähdesymboleja $a_1,\ldots,a_{N-1},a_N$ vastaamaan asetetaan välin $[0,1)$ osavälit

$$[0,p_1)\ ,\ [p_1,p_1+p_2)\ ,\ [p_1+p_2,p_1+p_2+p_3)\ ,\ldots,\ [p_1+\cdots+p_{N-1},1).$$

Merkitään a_i:tä vastaavaa väliä $[q_i,r_i)$:llä $(i = 1,\ldots,k)$. Lähdesymboli a_N on loppumerkki, ts. se esiintyy vain lähdesanan viimeisenä lähdesymbolina. Aritmeettisen koodauksen ideana on asettaa kutakin käypää lähdesanaa eli sellaista lähdesymbolien jonoa, jossa a_N esiintyy vain viimeisenä symbolina, vastaamaan yksikäsitteinen välin $[0,1)$ osaväli. Tämä tapahtuu seuraavasti:

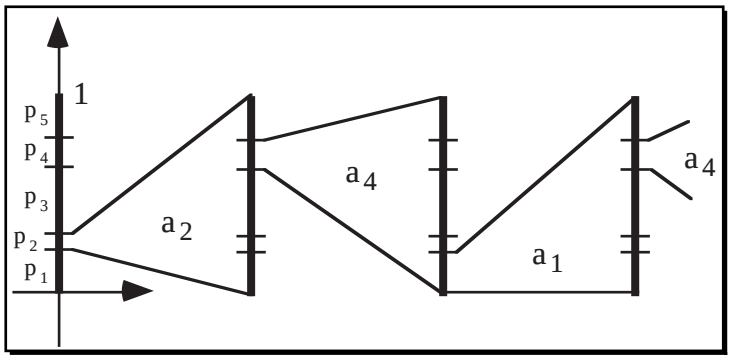
(1) Asetetaan $L \leftarrow 0$ ja $U \leftarrow 1$.

(2) Luetaan lähdesanan seuraava symboli $a = a_i$. Lasketaan

$$L^* = L + (U - L)q_i \qquad \text{ja} \qquad U^* = L + (U - L)r_i.$$

Huomaa, että $U^* - L^* = p_i(U - L)$.

(3) Jos $a = a_N$, tulostetaan $[L^*,U^*)$ ja lopetetaan. Muutoin asetetaan $L \leftarrow L^*$ ja $U \leftarrow U^*$ ja mennään kohtaan **(2)**.



On ilmeistä, että lähdesanaa tulee aina vastaamaan yksikäsitteinen väli $[L^*,U^*)$. Jos lähdesana on $\mathbf{u} = a_{i_1}a_{i_2}\cdots a_{i_n}$, niin $U^* - L^* = p_{i_1}p_{i_2}\cdots p_{i_n} = P(\mathbf{u})$.

Lähdesanan pituuden ei tässä kuitenkaan tarvitse olla kiinteä, mikä tahansa sana, jossa a_N esiintyy vain viimeisenä symbolina, käy.

Itse asiassa välin [L*,U*) asemasta riittää antaa yksikin välillä oleva reaaliluku x. Kun valitaan kokonaisluku R siten, että $10^{-R} \le U^* - L^* < 10^{-R+1}$, niin voidaan valita x:ksi luku $x = X/10^R$, missä X on välillä $[0,10^R)$ oleva kokonaisluku. Lähdesanaa **u** vastaavaksi koodisanaksi voidaankin ottaa X ja X:n pituus on enintään

$$R = \lceil -\lg (U^* - L^*) \rceil = \lceil -\lg P(\mathbf{u}) \rceil .$$

Jos käytetään n-pituisia lähdesanoja, niin koodisanan keskipituus on

$$\bar{\ell} \le \sum_{i=1}^{k} P_i \lceil -\lg P_i \rceil < -\sum_{i=1}^{k} P_i \lg P_i + 1 = H(U) + 1 \quad \text{(diteissä)},$$

kuten Shannonin koodauslauseen mukaan pitää ollakin. (Loppumerkin a_N todennäköisyydeksi otetaan tällöin $p_N = 1/n$ ja muut todennäköisyydet skaalataan vastaavasti.) Käytännössä $\bar{\ell}$ on kuitenkin useimmiten lähellä alarajaansa H(*U*) dittiä. Huomaa, että saadut sanat X eivät sellaisenaan muodosta (prefiksi)koodia ja ne pitää lähetettäessä erottaa.

Aritmeettisen koodauksen implementointi on ilmeisen vaativa tehtävä ja se myös vaikuttaa kriittisesti koodauksen “pakkauskykyyn”, ks. esimerkiksi WITTEN, I.H. & NEAL, R.M. & CLEARY, J.G.: Arithmetic Coding for Data Compression. *Communications of the Association for Computing Machinery* ***30*** (-87), 520–540. Mainittakoon, että ensimmäiset aritmeettisen koodauksen käytännön menetelmät kehitti suomalaissyntyinen informaatioteoreetikko Jorma Rissanen*.

7. Muistillinen lähde

Muistillisen lähteen tapauksessa lähteen tuottaman lähdesymbolin todennäköisyys riippuu sen aikaisemmin tuottamista symboleista, ts. lähde “muistaa”. Lähteen tuottaman symbolijonon voidaan ajatella alkaneen “äärettömän kauan sitten”, ts. se on muotoa

$$\ldots, u_{n-2}, u_{n-1}, u_n, \ldots$$

Lähdesymbolien jono muodostaa tällöin ns. stokastisen prosessin (ks. kurssi 73126 Stokastiset prosessit). Rajoittamalla tämä sopivaan tapaukseen (stationäärisyys, ergodisuus) voidaan todistaa eo. pykälien perustuloksia vastaavat tulokset myös muistillisille lähteille. Ks. esimerkiksi BLAHUT.

---

* RISSANEN, J.J.: Generalized Kraft Inequality and Arithmetic Coding. *IBM Journal of Research and Development* ***20*** (-76), 198–203.

IX LUKU

KANAVAKOODAUS

1. Muistiton kanava. Kanavan kapasiteetti

Palataan Luvun **VIII** tilanteeseen ja lisätään mukaan kanava, jonka kautta koodisanat lähetetään vastaanottajalle. Lähdeaakkoston A sekä koodiaakkoston C lisäksi tarvitaan siis vielä *vastaanottoaakkosto* $B = \{b_1,\ldots,b_L\}$.

Kanava on tässä kohiseva, ts. se voi aiheuttaa virheitä. Todennäköisyydet

$$P(\text{“vastaanotetaan } b_i\text{”}\,|\,\text{”lähetetään } c_j\text{”}) = p_{i\,|\,j} \quad (i = 1,\ldots,L; j = 1,\ldots,M)$$

oletetaan tunnetuksi. Kyseessä ovat ehdollisen tapausjärjestelmän

$$B\,|\,C : B_i\,|\,C_j \qquad (i = 1,\ldots,L; j = 1,\ldots,M),$$

missä $B_i = \{\text{“vastaanotetaan } b_i\text{”}\}$ ja $C_j = \{\text{”lähetetään } c_j\text{”}\}$ $(i = 1,\ldots,L; j = 1,\ldots,M)$, todennäköisyydet, jotka voidaan koota ehdolliseksi todennäköisyysmatriisiksi **Q** (ks. **VII.1**).

Kyseessä on tällöin ns. muistiton kanava. (Muistillisia kanavia ei tässä käsitellä.) Dekooderin tehtävä on paitsi “purkaa” koodaus, pyrkiä vähentämään kanavan aiheuttamien häiriöiden vaikutusta. Myös koodaus voidaan valita siten, että kanavan on mahdollisimman vaikea häiritä koodattua viestiä. Luvuissa **I–VI** käsitelty virheitä korjaava koodauskin pyritään suunnittelemaan kanavan ominaisuuksia ajatellen, mikäli mahdollista. Tämä saattaa vaikuttaa oleellisestikin koodin valintaan.

Kanava karakterisoidaan antamalla **Q**. Merkitään tällaista kanavaa K:llä. Jos merkitään täydellisen tapausjärjestelmän $B : B_1,\ldots,B_L$ todennäköisyysvektoria **p**:llä, tapausjärjestelmän $C : C_1,\ldots,C_M$ todennäköisyysvektoria **q**:lla sekä yhteistapausjärjestelmän $B \cap C$ todennäköisyysmatriisia **P**:llä, niin lähetetyn ja vastaanotetun symbolin välinen keskinäisinformaatio on (ks. Lause **32**)

$$I(B,C) = \mathrm{trace}(\mathbf{P}\log(\mathbf{Q}^{\mathrm{T}}\lceil\mathbf{p}\rfloor^{-1})) = \mathrm{trace}(\mathbf{Q}\lceil\mathbf{q}\rfloor\log(\mathbf{Q}^{\mathrm{T}}\lceil\mathbf{q}\mathbf{Q}^{\mathrm{T}}\rfloor^{-1})) = I(\mathbf{q}),$$

eli summamuodossa

$$I(q_1,\ldots,q_M) = \sum_{i=1}^{L}\sum_{j=1}^{M} p_{i|j}q_j \log \frac{p_{i|j}}{p_i}\,, \text{ missä } \quad p_i = \sum_{j=1}^{M} p_{i|j}q_j \quad (i = 1,\ldots,L).$$

Suurin mahdollinen I(**q**):n arvo on ns. kanavan K *kapasiteetti,* merkitään C(K). Kapasiteetti saadaan siis optimointitehtävän

$$\begin{cases} I(q_1,\ldots,q_M) = \max! \\ q_1 + \cdots + q_M = 1 \\ q_1 \geq 0,\ldots,q_M \geq 0 \end{cases}$$

ratkaisuna. Tehtävä ratkaistaan joko numeerisesti tai ns. Kuhn–Tucker-ehtoja käyttäen, ks. 73112 Matemaattinen optimointiteoria 1. Voidaan osoittaa, että kyseessä on ns. konveksi optimointitehtävä, mikä helpottaa ratkaisua. (Tähän tarvitaan Konveksisuuslausetta! Ks. esimerkiksi MCELIECE.)

2. Kanavakoodauslause

Merkitään nyt, kuten Luvussa **VIII**,

n = lähdesanan pituus,

m = koodisanan pituus,

$H(A)$ = lähteen entropia,

P_0 = dekoodausvirheen todennäköisyys.

Silloin saadaan (vrt. Shannonin lähteen koodauslause)

LAUSE 47. (SHANNONIN KANAVAKOODAUSLAUSE) Kaikille positiiviluvuille η ja π (miten tahansa pienille) on olemassa sellainen vakio $N_{\eta\pi}$, että jos $\frac{n}{m} H(A) \leq C(K) - \eta$, niin $P_0 < \pi$, kun koodaus ja dekoodaus valitaan sopivasti sekä $n > N_{\eta\pi}$.

Todistus. Ks. esimerkiksi YAGLOM & YAGLOM tai MCELIECE. ■

Myöskin vastaava “negatiivinen” tulos (vrt. Shannonin lähteen koodauslauseen osa **(ii)**) on olemassa, ks. YAGLOM & YAGLOM.

Entropian tulkinnan mukaan

$\frac{n}{m} H(A)$ = keskimääräinen lähdeinformaatio kanavaan lähetettyä symbolia kohti.

Shannonin kanavakoodauslauseen mukaan tämä voidaan saada mielivaltaisen lähelle arvoa C(K), jos sallitaan (häviävän) pieni virhetodennäköisyys. (Mainittu “negatiivinen” tulos sanoo suurin piirtein, että se ei voi ylittää arvoa C(K), ilman että virheen todennäköisyys jää merkittävän suureksi.)

3. Binäärinen symmetrinen kanava (BSC)

Binääriselle symmetriselle kanavalle B = C = {0,1} ja

$$\mathbf{Q} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}.$$

Näin ollen kumpikin virheistä $0 \leftarrow 1$ sekä $1 \leftarrow 0$ on kanavassa yhtä todennäköinen. Merkitään edelleen $\mathbf{q} = (q, 1-q)$. Silloin

$$\mathbf{p} = \mathbf{q}\mathbf{Q}^{\mathrm{T}} = (p + q - 2pq, 1 - p - q + 2pq),$$

$$\mathbf{P} = \mathbf{Q}\lceil \mathbf{q} \rfloor = \begin{pmatrix} (1-p)q & p(1-q) \\ pq & (1-p)(1-q) \end{pmatrix}$$

ja

$$\lceil \mathbf{p} \rfloor^{-1}\mathbf{Q} = \begin{pmatrix} \dfrac{1-p}{p+q-2pq} & \dfrac{p}{p+q-2pq} \\ \dfrac{p}{1-p-q+2pq} & \dfrac{1-p}{1-p-q+2pq} \end{pmatrix}.$$

Näin ollen (biteissä)

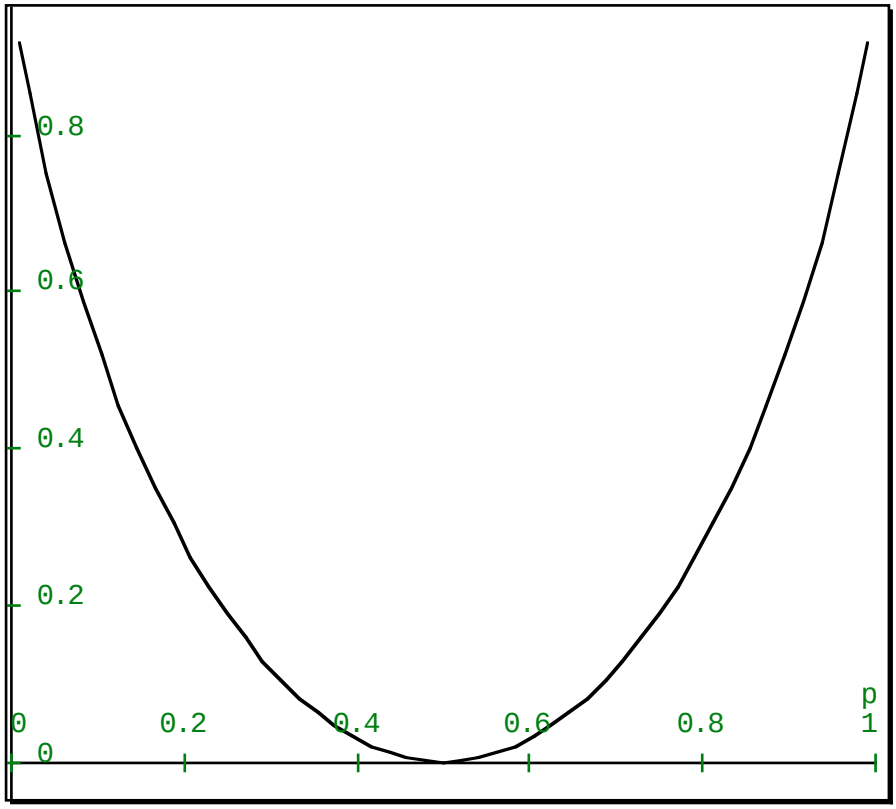
$$I = I(q,1-q) = (1-p)q \log_2 \frac{1-p}{p+q-2pq} + p(1-q) \log_2 \frac{p}{p+q-2pq}$$

$$+ pq \log_2 \frac{p}{1-p-q+2pq} + (1-p)(1-q) \log_2 \frac{1-p}{1-p-q+2pq}.$$

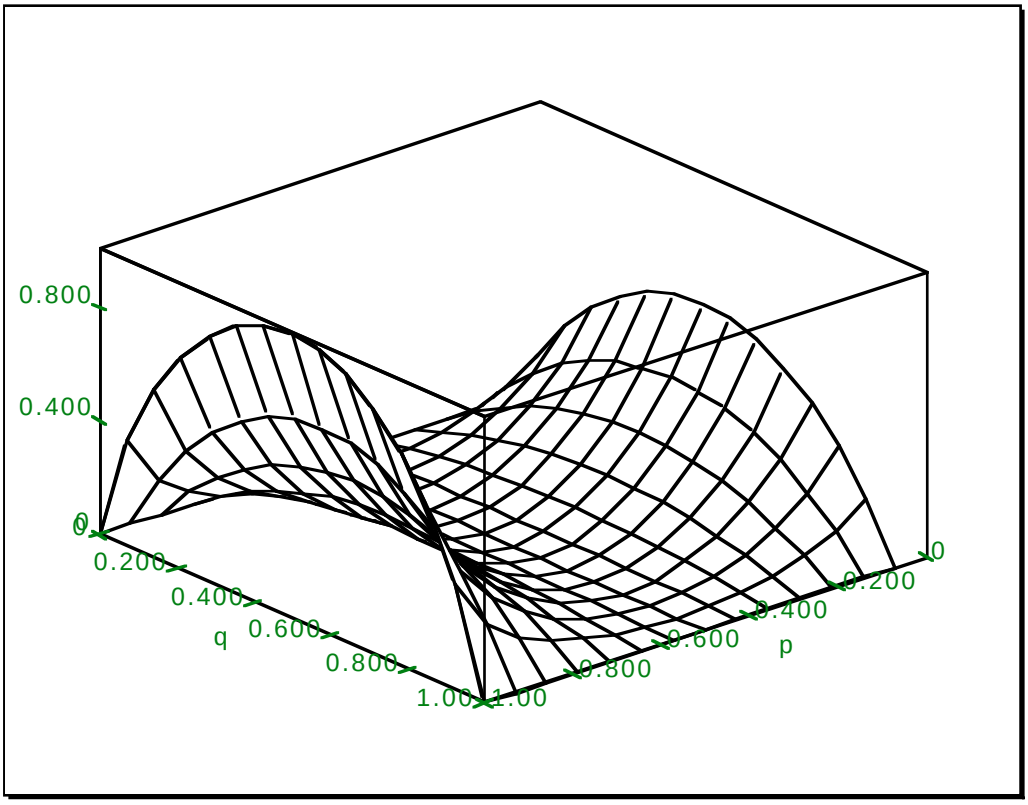
Jos erityisesti $p = 1/2$, niin $I = 0$, ts. kanava ei välitä informaatiota. I on symmetrinen pisteen $q = 1/2$ suhteen. Näin ollen kanavan kapasiteetti saavutetaankin, kun $q = 1/2$ ja

$$C(K) = 1 + (1-p) \log_2 (1-p) + p \log_2 p \quad \text{bittiä},$$

jonka kuvaaja (Maplella piirrettynä) on seuraavannäköinen:



I:n kuvaaja sekä p:n että q:n funktiona on seuraavanlainen (jälleen Maplella piirrettynä):



"The place that PME occupies in our present revolution is that it is one of the principles that has proved useful—and fairly general—in the task of developing that missing half (logically, the first half) of probability theory."

E.T. JAYNES (*SYNTHESE* ***63*** (-85))

X LUKU

MAKSIMIENTROPIAPERIAATE

1. Yleistä

Tilastotieteen kantavana periaatteena on mallintaa satunnaistilanne käyttäen tunnettuja parametrisoituja jakaumia ja estimoida parametrit otoksia käyttäen (ks. tilastomatematiikan kurssit). *Maksimientropiaperiaate* tarkoittaa jakauman valintaa tiettyjen ehtojen puitteissa siten, että jakauman entropia (differentiaalinen tai "tavallinen") maksimoituu pysyen äärellisenä. Jakaumatyyppiä ei tällöin kiinnitetä, vaan se on valittavissa vapaasti.

Entropian maksimoinnin perustana on seuraava ominaisuus:

APULAUSE. (i) (Diskreetti jakauma) Jos $\mathbf{p}$ ja $\mathbf{q}$ ovat todennäköisyysvektoreita (äärellisiä tai äärettömiä), niin

$$-\sum_i p_i \log p_i \le -\sum_i p_i \log q_i$$

ja yhtäläisyys on voimassa tarkalleen silloin, kun $\mathbf{p} = \mathbf{q}$.

(ii) (Jatkuva jakauma) Jos $f(\mathbf{x})$ ja $g(\mathbf{x})$ ovat tiheysfunktioita, niin

$$-\int_{-\infty}^{\infty} f(\mathbf{x}) \log f(\mathbf{x}) d\mathbf{x} \le -\int_{-\infty}^{\infty} f(\mathbf{x}) \log g(\mathbf{x}) d\mathbf{x}$$

ja yhtäläisyys on voimassa tarkalleen silloin, kun $f(\mathbf{x})$:n määrittämässä jakaumassa

$$P\big(f(\mathbf{x}) \neq g(\mathbf{x})\big) = 0.$$

Molemmissa tapauksissa oletetaan, että esiintyvät sarjat/integraalit suppenevat.

Todistus. **(i)** Voidaan olettaa, että $p_i \neq 0$ (jättämällä vastaavat tapaukset pois tapausjärjestelmästä). Silloin sivun 92 Apulauseen nojalla

$$\sum_i p_i \ln q_i - \sum_i p_i \ln p_i = \sum_i p_i \ln \frac{q_i}{p_i} \leq \sum_i p_i \left(\frac{q_i}{p_i} - 1 \right) \leq 0.$$

(ii) Integrointi voidaan rajoittaa alueeseen, jossa $f(\mathbf{x}) \neq 0$. Sen jälkeen todistus on samanlainen kuin kohdan **(i)**. ■

2. Maksimientropiajakaumat

Tapausjärjestelmää $S : A_1, A_2, \ldots$ (äärellistä tai ääretöntä) vastaten asetetaan diskreetti satunnaismuuttuja x seuraavasti: $x = i$, jos A_i tapahtuu.

LAUSE 48. (MAKSIMIENTROPIAMENETELMÄN PERUSLAUSE)

(i) (Diskreetti tapaus) Jos diskreetin satunnaismuuttujan x otosavaruus $\{1,\ldots,N\}$ tai $\{1,2,\ldots\}$ ja odotusarvo

$$E(\mathbf{g}(x)) = \sum_i p_i \mathbf{g}(i) = \underline{\theta},$$

missä $\mathbf{g} = (g_1,\ldots,g_n)$ ja $\underline{\theta} = (\theta_1,\ldots,\theta_n)$, on kiinnitetty, niin maksimientropiajakauma (mikäli olemassa) on muotoa

$$p_{MEi} = \gamma e^{-\underline{\lambda}\mathbf{g}(i)^T} \quad (i = 1,\ldots,N \text{ tai } i = 1,2,\ldots).$$

Vakiot γ ja $\underline{\lambda} = (\lambda_1,\ldots,\lambda_n)$ valitaan siten, että

$$\sum_i p_{MEi} = 1 \qquad \text{ja} \qquad \sum_i p_{MEi}\mathbf{g}(i) = \underline{\theta}.$$

(ii) (Jatkuva tapaus) Jos jatkuvan satunnaismuuttujan $\mathbf{x}$ otosavaruus Ω ja odotusarvo

$$E(\mathbf{g}(\mathbf{x})) = \int_\Omega f(\mathbf{x})\mathbf{g}(\mathbf{x})d\mathbf{x} = \underline{\theta},$$

missä $\mathbf{g} = (g_1,\ldots,g_n)$ ja $\underline{\theta} = (\theta_1,\ldots,\theta_n)$, on kiinnitetty, niin maksimientropiajakauma (mikäli olemassa) on muotoa

$$f_{ME}(\mathbf{x}) = \gamma e^{-\underline{\lambda}\mathbf{g}(\mathbf{x})^T} \quad (\mathbf{x} \in \Omega).$$

Vakiot γ ja $\underline{\lambda} = (\lambda_1,\ldots,\lambda_n)$ valitaan siten, että

$$\int_{\Omega} f_{ME}(\mathbf{x})d\mathbf{x} = 1 \quad \text{ja} \quad \int_{\Omega} f_{ME}(\mathbf{x})\mathbf{g}(\mathbf{x})d\mathbf{x} = \underline{\theta}.$$

Todistus. Näytetään vain kohta **(ii)** (kohdan **(i)** todistus on täysin analoginen). Lasketaan (nateissa)

$$\ln f_{ME}(\mathbf{x}) = \ln \gamma - \underline{\lambda}\mathbf{g}(\mathbf{x})^T$$

ja differentiaalientropia

$$H_{ME} = -\int_{\Omega} f_{ME}(\mathbf{x})\ln f_{ME}(\mathbf{x})d\mathbf{x} = -\int_{\Omega} f_{ME}(\mathbf{x})\left(\ln \gamma - \underline{\lambda}\mathbf{g}(\mathbf{x})^T\right)d\mathbf{x} = -\ln \gamma + \underline{\lambda}\underline{\theta}^T.$$

Sen näyttämiseksi, että kyseessä todella on maksimientropiajakauma, pitää osoittaa, että jos $f(\mathbf{x})$ on otosavaruuden Ω toinen tiheysfunktio, jolle odotusarvoehto $E(\mathbf{g}(\mathbf{x})) = \underline{\theta}$ toteutuu, niin sitä vastaava entropia ei ole suurempi kuin H_{ME}. Apulauseen nojalla mainittu entropia onkin

$$\begin{aligned} -\int_{-\underline{\infty}}^{\underline{\infty}} f(\mathbf{x})\ln f(\mathbf{x})d\mathbf{x} &\le -\int_{-\underline{\infty}}^{\underline{\infty}} f(\mathbf{x})\ln f_{ME}(\mathbf{x})d\mathbf{x} \\ &= -\int_{\Omega} f(\mathbf{x})\left(\ln \gamma - \underline{\lambda}\mathbf{g}(\mathbf{x})^T\right)d\mathbf{x} = -\ln \gamma + \underline{\lambda}\underline{\theta}^T = H_{ME} \end{aligned}$$

ja jakauma on myös yksikäsitteinen. ■

Vakioiden γ ja $\underline{\lambda}$ etsimiseksi saadaan $n + 1$ yhtälöä, joista ne (numeerisesti) ratkaistaan. (Tässä esimerkiksi Maple on hyvin kätevä apuväline.) On tietysti huomattava, ettei maksimientropiajakaumaa aina olekaan, ts. annettujen ehtojen puitteissa entropia saadaan miten tahansa suureksi.

Toisinaan on kätevää etsiä vakioiden $\underline{\lambda}$ arvot käyttäen ns. *ositusfunktiota*. Diskreetille jakaumalle ositusfunktio on

$$Z(\underline{\lambda}) = \sum_i e^{-\underline{\lambda}\mathbf{g}(i)^T} = \frac{1}{\gamma},$$

jolloin

$$-\frac{1}{Z}\frac{\partial Z}{\partial \lambda_j} = \gamma\sum_i g_j(i)e^{-\underline{\lambda}\mathbf{g}(i)^T} = \theta_j \quad (j = 1,\ldots,n).$$

Vastaavasti jatkuvan jakauman ositusfunktio on

$$Z(\underline{\lambda}) = \int_{\Omega} e^{-\underline{\lambda}\mathbf{g}(\mathbf{x})^T}d\mathbf{x} = \frac{1}{\gamma}$$

ja samoin $-\frac{1}{Z}\frac{\partial Z}{\partial \lambda_j} = \theta_j$ (j = 1,...,n). Näin saadaan n yhtälöä tuntemattomille $\lambda_1,\ldots,\lambda_n$.

HUOM! Valitsemalla g siten, että

$$g(x) = \begin{cases} c_1, \text{ kun } a_1 \le x \le b_1 \\ c_2, \text{ kun } a_2 \le x \le b_2 \\ \quad\vdots \\ c_k, \text{ kun } a_k \le x \le b_k \end{cases}$$

missä $a_1 < b_1 < a_2 < b_2 < \cdots < a_k < b_k$, saadaan lausuttua välien $[a_1,b_1],[a_2,b_2],\ldots,[a_k,b_k]$ todennäköisyyksiä koskevia ehtoja.

Peruslauseen antamat jakaumat ovat ns. *eksponentiaalityyppisiä* jakaumia. Niiden maksimientropiaominaisuus selittää näiden jakaumien esiintymisen niin usein tilastotieteessä, luotettavuusteoriassa jms.:

ESIMERKKI 1. Jos $\Omega = [a,b]$ ja lisäehtoja ei ole (tai valitaan n = 1, g(x) = 0 ja $\theta = 0$), saadaan

$$f_{ME}(x) = \gamma e^{-\lambda \cdot 0} = \gamma, \text{ kun } a \le x \le b,$$

ts. kyseessä on tasajakauma välille [a,b].

ESIMERKKI 2. Jos $\Omega = (-\infty,\infty)$ ja lisäehtoja ei ole, maksimientropiajakaumaa ei selvästikään ole. Myöskään yo. menettely ei silloin tuota tulosta.

ESIMERKKI 3. Jos $\Omega = [0,\infty)$ ja g(x) = x, niin

$$f_{ME}(x) = \gamma e^{-\lambda x}, \text{ kun } x \ge 0,$$

ts. maksimientropiajakauma on eksponenttijakauma.

ESIMERKKI 4. Jos $\Omega = (-\infty,\infty)$ ja $g_1(x) = x$ ja $g_2(x) = x^2$, niin maksimientropiajakauma on normaalijakauma $N(\theta_1,\theta_2 - \theta_1^2)$, sillä

$$f_{ME}(x) = \gamma e^{-\lambda_1 x - \lambda_2 x^2}.$$

ESIMERKKI 5. Jos äärettömälle diskreetille jakaumalle g(x) = x, niin

$$p_{MEi} = \gamma e^{-\lambda i} \qquad (i = 1,2,\ldots),$$

ts. maksimientropiajakauma on geometrinen jakauma.

Kirjallisuutta

ADÁMEK, J.: *Foundations of Coding. Theory and Applications of Error-Correcting Codes with an Introduction to Cryptography and Information Theory*. Wiley (–91)

ANDERSON, J.B. & MOHAN, S.: *Source and Channel Coding. An Algorithmic Approach*. Kluwer (–91)

BLAHUT, R.E.: *Theory and Practice of Error Control Codes*. Addison–Wesley (–83)

BLAHUT, R.E.: *Principles and Practice of Information Theory*. Addison–Wesley (–87)

HEISE, W. & QUATTROCCHI, P.: *Informations- und Codierungstheorie*. Springer–Verlag (–95)

HOFFMAN, D.G. & LEONARD, D.A. LINDNER, C.C. & PHELPS, K.T. & RODGER, C.A. & WALL, J.R.: *Coding Theory. The Essentials*. Dekker (–91)

LIN, S. & COSTELLO JR., D.J.: *Error Control Coding. Fundamentals and Applications*. Prentice–Hall (–83)

MACWILLIAMS, F.J. & SLOANE, N.J.A.: *The Theory of Error-Correcting Codes*. North–Holland (–78)

MCELIECE, R.J. & ASH, R.B. & ASH, C.: *Introduction to Discrete Mathematics*. McGraw–Hill (–89)

MCELIECE, R.J.: *Finite Fields for Computer Scientists and Engineers*. Kluwer (–87)

MCELIECE, R.J.: *The Theory of Information and Coding. A Mathematical Framework for Communication*. Addison–Wesley (–83)

PAPOULIS, A.: *Probability & Statistics*. Prentice–Hall (–90)

RAO, T.R.N. & FUJIWARA, E.: *Error-Control Coding for Computer Systems*. Prentice–Hall (–89)

SWEENEY, P.: *Error Control Coding. An Introduction*. Prentice–Hall (–91)

VAN LINT, J.H.: *Introduction to Coding Theory*. Springer–Verlag (–92)

VANSTONE, S.A. & VAN OORSCHOT, P.C.: *An Introduction to Error Correcting Codes with Applications*. Kluwer (–89)

VITERBI, A.J. & OMURA, J.K.: *Digital Communication and Coding*. McGraw–Hill (–79)

YAGLOM, A.M. & YAGLOM, I.M.: *Probability and Information*. Reidel (–83)

Hakemisto