Survo-ristikon ratkaiseminen matriiseilla

Kimmo Vehkalahti

9.6.2011

1 Taustaa

Harrastettuani **Survo-ristikoita** tiiviisti keväästä 2006 lähtien, toisin sanoen siitä asti, kun *Seppo Mustonen* tuon koukuttavan ajanvietteen keksi, sain keväällä 2007 idean **matriiseihin perustuvasta ratkaisutavasta**, jossa saatoin hyödyntää monia Survo-ohjelmiston toimintoja, ennen kaikkea sen **mainiota matriisitulkkia**.

Työstin ratkaisutapaani varsin pitkälle kesällä 2007, mutta syksyn tullen työt alkoivat haitata harrastuksia, ja jouduin jättämään ristikoiden pohtimisen sikseen. Ainoaksi dokumentiksi jäi luonnosmainen selostukseni[1] huhtikuulta 2007. (Laitoin onnekseni muistiinpanojani ja kokeilujani talteen lukuisiin Survon toimituskenttiin.)

Eräät ratkaisutapaani koskevat kysymykset jäivät pyörimään päähäni, ja yritin silloin tällöin palata niihin, mutta se onnistui vasta yli kolme vuotta myöhemmin, syksyllä 2010. Muutamien yksittäisten pohdintojen tuloksena innostuin ja perehdyin asiaan uudelleen joululomalla. Vuoden vaihduttua aloin laatia uutta dokumenttia, jossa keksin lähestyä asiaa pienimmän mahdollisen ristikon avulla.

2 Pienen Survo-ristikon ratkaiseminen

Survo-ristikon koon määräävät sen vaaka- ja pystyrivien lukumäärät m ja n. Erikoistapauksina voidaan ajatella "ristikoita", joissa $m = 1$ tai $n = 1$, mutta käytännössä kiinnostavat vain ristikot, joissa $m > 1$ ja $n \geq m$ (tai toisinpäin). Erikoistapausten ja kiinnostavien ristikoiden "väliin" jää ainostaan tapaus, jossa $m = 2$ ja $n = 2$. Yleensä niin pieni Survo-ristikko ei kiinnosta lainkaan, mutta nyt se osoittautui käteväksi ratkaisutapani havainnollistamisessa.

Näytän yksityiskohtaisesti, miten alla näkyvä avoin 2×2 -ristikko ratkeaa matriisien avulla. Ratkaisu näkyy keskellä. Kolmas ei ole kelvollinen Survo-ristikko, koska sen ratkaisu ei ole yksikäsitteinen. Siihen palataan kuitenkin kohdassa 2.5.

	A	B	
1			3
2			7
	4	6	

	A	B	
1	1	2	3
2	3	4	7
	4	6	

	A	B	
1	1	2	3
2	4	3	7
	5	5	

Kaikki olennainen näistä kahdesta ristikosta on todettu aivan ensimmäisessä aihepiirin selostuksessa (Mustonen 2006a, 12). Uutta on nyt se, miten ratkaisuihin päädytään erilaisten matriisien ja matriisioperaatioiden avulla.

[1] `http://www.survo.fi/ristikot/Kimmo_53_2007.pdf`

2.1 Ositukset binäärimuodossa

Survo-ristikoiden ratkaisemisessa tarvitaan tietoa siitä, mitkä luvut voivat tulla kyseeseen millekin vaaka- ja pystyriville, kun kukin luvuista $1, 2, \ldots, mn$ saa esiintyä ristikossa vain kerran. Lukujen järjestys määräytyy yhdistelmänä ennalta annettujen vaaka- ja pystyrivisummien osituksista eli *partitioista* P, joita on vähintään yksi jokaista vaaka- ja pystyriviä kohden (Mustonen 2006a).

Nyt käsiteltävässä 2×2 -ristikossa tilanne on niin yksinkertainen, ettei osituksille ole vaihtoehtoja, vaan $P_1 = \{1, 2\}$, $P_2 = \{3, 4\}$, $P_A = \{1, 3\}$ ja $P_B = \{2, 4\}$.

Ratkaisutapani **keskeinen idea** on tarkastella osituksia *binäärimuodossa*. Siinä jokainen ositus kuvataan mn-pituisena sarjana nollia ja ykkösiä, jossa ykköset ilmaisevat luvun kuulumisen kyseiseen ositukseen. Esimerkissä luvut 1 ja 2 kuuluvat ositukseen P_1, kun taas luvut 3 ja 4 eivät:

	1	2	3	4
P_1	1	1	0	0
P_2	0	0	1	1
P_A	1	0	1	0
P_B	0	1	0	1

Siirtymällä matriisinotaatioon ja merkitsemällä osituksia 1×4 -kokoisina vektoreina $\boldsymbol{P}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$, $\boldsymbol{P}_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$, $\boldsymbol{P}_A = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$ ja $\boldsymbol{P}_B = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$ päästään alkuun matriiseilla operoinnissa. Talletetaan nämä vektorit myös Survon matriisitulkilla käsiteltäviksi:

```
*MAT SAVE AS P1
*1 1 0 0
*
*MAT SAVE AS P2
*0 0 1 1
*
*MAT SAVE AS PA
*1 0 1 0
*
*MAT SAVE AS PB
*0 1 0 1
```

Kun osituksia on enemmän, nämä ovat $p_i \times mn$ -matriiseja, jossa p_i on kyseisen (vaaka- tai pysty)rivin ositusten lukumäärä, kun $i = 1, 2, \ldots, mn$.

2.2 Ositukset koodattuna

Ristikon ratkaiseminen matriiseilla perustuu edellä muodostettujen, binäärimuotoisten ositusten parittaisiin tarkasteluihin. Kerralla tarkastellaan **yhden vaaka- ja yhden pystyrivin** osituksia. Sovellettavan koodauksen ansiosta tulee joka vaiheessa huomioiduksi koko ristikon tilanne. Kun jatketaan vaiheittain, päästään lopulta ristikon yksikäsitteiseen ratkaisuun (olettaen, että sellainen ylipäätään on olemassa).

Aloitetaan järjestyksessä vaakarivistä 1 ja pystyrivistä A, siis matriiseista

$$\boldsymbol{P}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \quad \text{ja} \quad \boldsymbol{P}_A = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}.$$

Matriiseista $\boldsymbol{P}_1$ ja $\boldsymbol{P}_A$ nähdään, että molempien ensimmäinen elementti on ykkönen, kaksi seuraavaa menevät keskenään ristiin ja viimeinen on molemmissa nolla. Se tarkoittaa ensinnäkin sitä, että riveillä 1 ja A on tasan yksi yhteinen luku, kuten pitääkin. Molemmilla on myös yksi oma luku. Lisäksi ristikkoon sisältyy yksi luku, joka ei kuulu kumpaankaan näistä riveistä.

Seuraavaksi rivikohtaiset binäärimuotoiset tiedot **koodataan uudelleen** *kertomalla toinen matriiseista kahdella*, minkä jälkeen matriisit *lasketaan yhteen*:

$$\boldsymbol{P}_{1A} = \boldsymbol{P}_1 + 2\boldsymbol{P}_A = \begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix} + \begin{bmatrix}2 & 0 & 2 & 0\end{bmatrix} = \begin{bmatrix}3 & 1 & 2 & 0\end{bmatrix}.$$

Ilman uudelleenkoodausta yhteenlasku tuottaisi matriisin $\begin{bmatrix}2 & 1 & 1 & 0\end{bmatrix}$, jossa rivien omilla elementeillä olisi sama koodi (ykkönen). Nyt saatu matriisi $\boldsymbol{P}_{1A}$ sisältää yksikäsitteisen koodin jokaiselle neljälle elementille:

- koodi 3 (vaakarivin 1 ja pystyrivin A yhteinen luku)
- koodi 1 (vaakarivin 1 oma luku)
- koodi 2 (pystyrivin A oma luku)
- koodi 0 (vaakarivin 1 ja pystyrivin A ulkopuolinen luku)

Survo-ristikkoon sijoitettuna tilanne näyttää seuraavalta (rivisummat on jätetty pois ja koodit 0–3 kirjoitettu harmaalla erotukseksi ristikon varsinaisista luvuista):

	A	B
1	3	1
2	2	0

	A	B	C
1	3	1	1
2	2	0	0
3	2	0	0

	A	B	C
1	0	2	0
2	1	3	1
3	0	2	0

Tähänastinen riittää jo 2×2 -ristikon ratkaisemiseen, sillä jokaista ristikkoon sijoitettavaa lukua (1, 2, 3, 4) vastaa yksikäsitteinen koodi (3, 1, 2, 0). Kun ristikon koko kasvaa, osa koodeista muuttuu monikäsitteisiksi. Samat neljä koodia toimivat silti, mikä näkyy yllä kahdesta 3×3 -ristikosta. Niiden keskinäisenä erona on vain se, että toisessa on tarkasteltu rivejä 1 ja A, kun taas toisessa rivejä 2 ja B. Tähän palataan tarkemmin seuraavassa kohdassa. Yleisesti $m \times n$ -ristikon tilanteessa **tarvittavat neljä koodia** ja niiden lukumäärät ovat:

- koodi 3 (vaakarivin ja pystyrivin yhteinen luku), 1 kpl
- koodi 1 (vaakarivin omat luvut), $n - 1$ kpl
- koodi 2 (pystyrivin omat luvut), $m - 1$ kpl
- koodi 0 (vaakarivin ja pystyrivin ulkopuoliset luvut), $(n - 1)(m - 1)$ kpl

Ristikko ratkeaa, kun sen jokaista lukua vastaa yksikäsitteinen koodi. Tähän pääsemiseksi koodausta on laajennettava.

2.3 Koodaukset yhdistettynä

Edellä oikeanpuoleisessa 3×3 -ristikossa on siirrytty tarkastelemaan rivejä 2 ja B. Vastaava tarkastelu ei 2×2 -tilanteessa ole välttämätöntä, koska ristikko ratkeaa jo ensimmäisessä vaiheessa. Tehdään se kuitenkin havainnollisuuden vuoksi, jotta päästään pohtimaan koodauksen laajentamista. Tuloksena saatava koodaus riittää jo 3×3 -ristikon ratkaisemiseen.

Aiemmin esitetyn perusteella saadaan suoraviivaisesti

$$\boldsymbol{P}_{2B} = \boldsymbol{P}_2 + 2\boldsymbol{P}_B = \begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix} + \begin{bmatrix}0 & 2 & 0 & 2\end{bmatrix} = \begin{bmatrix}0 & 2 & 1 & 3\end{bmatrix}.$$

Tämän jälkeen olisi yhdistettävä kaksi samalla tavalla koodattua matriisia,

$$\boldsymbol{P}_{1A} = \begin{bmatrix} 3 & 1 & 2 & 0 \end{bmatrix} \text{ ja } \boldsymbol{P}_{2B} = \begin{bmatrix} 0 & 2 & 1 & 3 \end{bmatrix},$$

mutta siihen edellä esitetty yksinkertainen koodaustapa ei sovellu. On helppo nähdä, että se johtaisi monikäsitteisiin koodeihin.

Koodausten yhdistämiseksi matriisit $\boldsymbol{P}_{1A}$ ja $\boldsymbol{P}_{2B}$ **koodataan uudelleen sopivilla muunnoksilla**. Tällaisiksi muunnoksiksi voidaan valita esimerkiksi $2x + 1$ ja 2^x, joissa x viittaa yleisesti muunnettavan matriisin alkioon. Näillä muunnoksilla uusiksi matriiseiksi saadaan

$$\boldsymbol{Q}_{1A} = \begin{bmatrix} 7 & 3 & 5 & 1 \end{bmatrix} \text{ ja } \boldsymbol{Q}_{2B} = \begin{bmatrix} 1 & 4 & 2 & 8 \end{bmatrix},$$

jotka *kerrotaan keskenään alkioittain*, eli muodostetaan niiden *Hadamardin tulo*[2]

$$\boldsymbol{Q}_{1A2B} = \boldsymbol{Q}_{1A} \circ \boldsymbol{Q}_{2B} = \begin{bmatrix} 7 & 12 & 10 & 8 \end{bmatrix}.$$

Tähänastisia vaiheita Survon matriisitulkilla:

```
*MAT P1A=P1+2*PA  /  yhdistetty koodaus: {0,1,2,3}
*MAT P2B=P2+2*PB  /  yhdistetty koodaus: {0,1,2,3}
*
*MAT Q1A=P1A
*MAT Q2B=P2B
*MAT TRANSFORM Q1A BY 2*X#+1  /  uudelleenkoodaus: {1,3,5,7}
*MAT TRANSFORM Q2B BY 2^X#    /  uudelleenkoodaus: {1,2,4,8}
*
*MAT Q1A2B=#HADAMARD(Q1A,Q2B) /  Hadamardin tulo (ks. kohta 4.4)
```

Yhdistetyn koodauksen hyöty selviää varsinaisesti vasta tarkasteltaessa 3×3 -ristikkoa. Alla ovat matriiseja $\boldsymbol{Q}_{1A}$, $\boldsymbol{Q}_{2B}$ ja $\boldsymbol{Q}_{1A2B}$ vastaavat vaiheet koodattuina 3×3 -ristikoina:

	A	B	C
1	7	3	3
2	5	1	1
3	5	1	1

	A	B	C
1	1	4	1
2	2	8	2
3	1	4	1

	A	B	C
1	7	12	3
2	10	8	2
3	5	4	1

Matriisin $\boldsymbol{Q}_{1A2B}$ myötä on siis aikaansaatu laajennettu koodaus, jolla hallitaan jo 3×3 -ristikko täydellisesti. Sen sijaan 4×4 -tilanteeseen se ei vielä riitä:

	A	B	C	D
1	7	12	3	3
2	10	8	2	2
3	5	4	1	1
4	5	4	1	1

Vasen ylänurkka pysyy ennallaan, mutta lisäriveissä 4 ja D toistuvat koodit 1–5. Koodausta on siis laajennettava edelleen.

[2]Jacques Hadamard 1865–1963, ks. `http://fi.wikipedia.org/wiki/Jacques_Hadamard`

2.4 Koodausristikon muodostaminen

Koodausta laajennettaessa on huolehdittava, etteivät uudet koodit sekoitu aiempiin. Ykkönen – käsiteltävien rivien ulkopuolisten lukujen koodi – sisältyy joka muunnokseen, muissa on vaihtelua. Lopulliseen ”koodausristikkoon” pätee siis sama sääntö kuin Survo-ristikkoon: kukin luku saa esiintyä vain kerran.

Koodausristikot voi tehdä suoraan ristikoita vastaavilla matriiseilla. Seuraavassa rakennetaan Survon matriisitulkilla 4×4 -koodausristikko:

```
*/ACTIVATE +  /  tämän aktivointi aktivoi kaikki rivit, joiden kontrollimerkkinä on +
*
+MAT X=ZER(4,4)  /  Muodostetaan 4 x 4 -nollamatriisi X ja
+MAT X(0,1)="A"  /  otsikoidaan se Survo-ristikon tapaan.
+MAT X(0,2)="B"
+MAT X(0,3)="C"
+MAT X(0,4)="D"
+MAT RLABELS NUM(1) TO X
*
*Otetaan X otsikoineen pohjaksi rivien binäärimuodoille:
*
+MAT X1=X  / Kolme vaaka- ja pystyriviparia riittää:
+MAT XA=X  / valitaan X1 ja XA, X2 ja XB, X3 ja XC.
+MAT X2=X  /
+MAT XB=X  /                                              Tältä näyttävät matriisit
+MAT X3=X  /                                              ennen uudelleenkoodauksia:
+MAT XC=X  /
*                                                        X1:      X2:      X3:
*Binäärikoodataan matriiseihin tieto vast. rivistä:      1 1 1 1  0 0 0 0  0 0 0 0
*(I# viittaa matriisin rivi- ja J# sarakeindeksiin)      0 0 0 0  1 1 1 1  0 0 0 0
*                                                        0 0 0 0  0 0 0 0  1 1 1 1
+MAT #TRANSFORM X1 BY F1 / F1=if(I#=1)then(1)else(0)     0 0 0 0  0 0 0 0  0 0 0 0
+MAT #TRANSFORM XA BY FA / FA=if(J#=1)then(1)else(0)
+MAT #TRANSFORM X2 BY F2 / F2=if(I#=2)then(1)else(0)     XA:      XB:      XC:
+MAT #TRANSFORM XB BY FB / FB=if(J#=2)then(1)else(0)     1 0 0 0  0 1 0 0  0 0 1 0
+MAT #TRANSFORM X3 BY F3 / F3=if(I#=3)then(1)else(0)     1 0 0 0  0 1 0 0  0 0 1 0
+MAT #TRANSFORM XC BY FC / FC=if(J#=3)then(1)else(0)     1 0 0 0  0 1 0 0  0 0 1 0
*                                                        1 0 0 0  0 1 0 0  0 0 1 0
*Yhdistelyt ja uudelleenkoodaukset:
*                                                        X1A:     X2B:     X3C:
+MAT X1A=X1+2*XA         / W=if(X#=0)then(01)else(a)     3 1 1 1  0 2 0 0  0 0 2 0
+MAT X2B=X2+2*XB         / a=if(X#=1)then(09)else(b)     2 0 0 0  1 3 1 1  0 0 2 0
+MAT X3C=X3+2*XC         / b=if(X#=2)then(11)else(c)     2 0 0 0  0 2 0 0  1 1 3 1
*U=2*X#+1 V=2^X#         / c=if(X#=3)then(13)else(X#)    2 0 0 0  0 2 0 0  0 0 2 0
*
+MAT #TRANSFORM X1A BY U / U: {1,3,5,7}
+MAT #TRANSFORM X2B BY V / V: {1,2,4,8}
+MAT #TRANSFORM X3C BY W / W: {1,9,11,13}
*                                                        Alla matriisit X1A, X1B, X1C
*                                                        uudelleenkoodausten jälkeen:
*
*Yhdistelyt Hadamardin tulona X1A o X2B o X3C:        /      X1A:     X2B:     X3C:
*                                                     /      7 3 3 3  1 4 1 1  1 1 11 1
+MAT        X1A2B=#HADAMARD(X1A,X2B)                  /      5 1 1 1  2 8 2 2  1 1 11 1
+MAT      X1A2B3C=#HADAMARD(X1A2B,X3C)                /      5 1 1 1  1 4 1 1  9 9 13 9
+MAT NAME X1A2B3C AS 4x4                              /      5 1 1 1  1 4 1 1  1 1 11 1
*
+LOADM X1A2B3C 111 CUR+2  /  valmis koodausristikko
*
*4x4
*           A   B   C   D
*  1        7  12  33   3
*  2       10   8  22   2
*  3       45  36  13   9
*  4        5   4  11   1
*
```

2.5 Yleistäminen usean osituksen tilanteeseen

Yleistetään nyt yksinkertaisinta tilannetta siten, että **mahdollisia osituksia saa olla useampia kuin yksi.** Tutkitaan sivulla 1 esiintynyttä ristikkoa, jolla ei ole yksikäsitteistä ratkaisua, ja käydään sen avulla läpi samat vaiheet kuin edellä.

2.5.1 Ositukset binäärimuodossa

Todetaan tilanne Survon COMB-komennolla ja talletetaan vastaavat binääritiedot kuin alussa. Ero on nyt se, että kahdessa matriiseista rivejä on enemmän kuin yksi:

```
*                 A B
* Käsiteltävä   1 1 2 3
* ristikko on   2 4 3 7
*                 5 5
*
*Pystyrivien A ja B summalla 5 on kaksi ositusta: {1,4} ja {2,3}:
*
*COMB P CUR+1    /  P=PARTITIONS,5,2 DISTINCT=1 MIN=1 MAX=4
*Partitions 2 of 5: N[P]=2
*1 4
*2 3
*
*MAT SAVE AS P1  /  (sama kuin alussa)
*1 1 0 0
*
*MAT SAVE AS P2  /  (sama kuin alussa)
*0 0 1 1
*
*MAT SAVE AS PA  /  tässä on ratkaiseva ero...
*1 0 0 1
*0 1 1 0
*
*MAT SAVE AS PB  /  ...samoin kuin tässä
*0 1 1 0
*1 0 0 1
```

Edellä esitetyn mukaisesti näistä käytetään myös merkintöjä $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_A$ ja $\boldsymbol{P}_B$.

2.5.2 Ositukset koodattuna

Ero aiempaan seuraa välittömästi, sillä $\boldsymbol{P}_1$ ja $\boldsymbol{P}_A$ ovat eri kokoisia:

```
*MAT DIM P1 /* rowP1=1 colP1=4
*MAT DIM PA /* rowPA=2 colPA=4
```

Samasta syystä ei myöhemmässä yhdistelyvaiheessa voida myöskään hyödyntää suoraan Hadamardin tuloa. **Kaikki vaiheet on yleistettävä** tilanteisiin, joissa matriisien dimensiot vaihtelevat rivien ositusten määrien mukaan.

Joustava keino matriisien yhdistelyyn niiden dimensioista riippumatta on nimeltään *Kroneckerin tulo*[3]. Se on lopulta ratkaisutapani **todellinen avain**, ja tulee käyttöön sitä useammassa vaiheessa, mitä pidemmälle ratkaisussa edetään.

Kroneckerin tulolla voidaan yhdistää mitkä tahansa kaksi matriisia, esimerkiksi $m \times n$ -matriisi $\boldsymbol{A}$ ja $p \times q$ -matriisi $\boldsymbol{B}$. Tuloksena saadaan $mp \times nq$ -matriisi

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix},$$

jossa $\boldsymbol{A}$:n jokaisen luvun a_{ij} tilalla on ko. luvulla kerrottu $\boldsymbol{B}$.

[3]Leopold Kronecker 1823–1891, ks. `http://fi.wikipedia.org/wiki/Leopold_Kronecker`

Matriisien $\boldsymbol{P}_1$ ja $2\boldsymbol{P}_A$ yhdistäminen Kroneckerin tulolla onnistuu, koska dimensiot saavat olla mitä tahansa, mutta lopputulos ei tyydytä, koska koodaus hajoaa:

```
*MAT K=KRONECKER(P1,2*PA)
*MAT LOAD K 111 CUR+1
*MATRIX K
*KRONECKER(P1,2*PA)
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1        2   0   0   2   2   0   0   2   0   0   0   0   0   0   0   0
*1*2        0   2   2   0   0   2   2   0   0   0   0   0   0   0   0   0
```

Yhdistämisessä tarvittava **yhteenlasku** mahdollistuu, kun muokataan matriisit *yhteenlaskukelpoisiksi* Kroneckerin tulon avulla. Lisäksi tarvitaan peräti kolmea eri kokoista "ykköstolppaa", siis vektoria, joissa on pelkkiä ykkösiä:[4]

```
*MAT One1=CON(rowP1,1)    /  dimensiot P1:n ja PA:n mukaisesti,
*MAT OneA=CON(rowPA,1)    /   symboliset dimensiomerkinnät saadaan
*MAT Ones=CON(colP1,1)    /   edellä olleilla MAT DIM -komennoilla
*MAT NAME One1 AS 1
*MAT NAME OneA AS 1       /  selvyyden vuoksi sisäisiksi nimiksi
*MAT NAME Ones AS 1       /   pannaan kaikkiin näihin pelkkä ykkönen
*
*MAT LOAD One1 111 CUR+1  /  skalaari, koska rivillä 1 vain 1 ositus
*MATRIX One1
*1
*///        1
*  1        1
*
*MAT LOAD OneA 111 CUR+1  /  rivillä A sen sijaan on 2 ositusta
*MATRIX OneA
*1
*///        1
*  1        1
*  2        1
*
*MAT LOAD Ones' 111 CUR+1 /  myös tällaista transpoosia tarvitaan
*MATRIX Ones
*1'
*///        1   2   3   4
*  1        1   1   1   1
```

Ykköstolpista rakennetaan kaksi pelkistä ykkösistä koostuvaa matriisia, joiden rividimensiot vastaavat kääntäen matriisien $\boldsymbol{P}_1$ ja $\boldsymbol{P}_A$ dimensioita ja joiden sarakedimensio on ristikkoon tulevien lukujen määrä eli 4:

```
*MAT Ones1=One1*Ones'
*MAT OnesA=OneA*Ones'
*
*MAT LOAD Ones1 111 CUR+1
*MATRIX Ones1
*1*1'
*///        1   2   3   4
*  1        1   1   1   1
*
*MAT LOAD OnesA 111 CUR+1
*MATRIX OnesA
*1*1'
*///        1   2   3   4
*  1        1   1   1   1
*  2        1   1   1   1
```

Kun alun perin eri kokoisiin matriiseihin $\boldsymbol{P}_1$ ja $2\boldsymbol{P}_A$ sovelletaan Kroneckerin tuloa vastaavien ykkösmatriisien kanssa, saadaan samankokoiset (2×16) matriisit:

[4]Tässä kohdassa ratkaisevan neuvon minulle antoi *Simo Puntanen*, ykköstolppien erikoismies.

```
*MAT K1=KRONECKER(P1,OnesA)
*MAT KA=KRONECKER(Ones1,2*PA)
*
*MAT LOAD K1 111 CUR+1
*MATRIX K1
*KRONECKER(P1,1*1')
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1       1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0
*1*2       1   1   1   1   1   1   1   1   0   0   0   0   0   0   0   0
*
*MAT LOAD KA 111 CUR+1
*MATRIX KA
*KRONECKER(1*1',2*PA)
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1       2   0   0   2   2   0   0   2   2   0   0   2   2   0   0   2
*1*2       0   2   2   0   0   2   2   0   0   2   2   0   0   2   2   0
```

Nämä eivät yksin auta (vrt. aiempi kokeilu), mutta yhteenlasku onnistuu:

```
*MAT K=K1+KA
*MAT LOAD K 111 CUR+1
*MATRIX K
*KRONECKER(P1,1*1')+KRONECKER(1*1',2*PA)
*///      1*1 1*2 1*3 1*4 2*1 2*2 2*3 2*4 3*1 3*2 3*3 3*4 4*1 4*2 4*3 4*4
*1*1       3   1   1   3   3   1   1   3   2   0   0   2   2   0   0   2
*1*2       1   3   3   1   1   3   3   1   0   2   2   0   0   2   2   0
```

Päästiin edeltä tuttuun {0,1,2,3}-koodaukseen. **Lukuja vain on liikaa!**

Kroneckerin tulo operoi molempiin suuntiin, mutta ratkaisutavassani kiinnostaa ainoastaan rivisuunta, koska se yhdistelee ositukset. Sarakkeiden yhdistelyistä kiinnostavia ovat vain neljä Hadamardin tuloa vastaavaa "pääsaraketta", jotka Survon matriisitulkki otsikoi edellä 1*1, 2*2, 3*3 ja 4*4.

Kroneckerin tulon monipuolisuutta kuvaa se, että kiinnostavat sarakkeet saadaan erotettua äskeisestä sen avulla muodostetulla valintamatriisilla. Kun edellä yhdistelymatriiseissa esiintyi pelkkiä ykkösiä, on valintamatriisissa enimmäkseen nollia:

```
*MAT I=IDN(colP1,colP1)            /  yksikkömatriisi (4 x 4)
*MAT One=CON(colP1,1)              /   ja ykköstolppa (4 x 1)
*
*MAT J1=KRONECKER(I,One)           /  Kroneckerin tulo 1,
*MAT J2=KRONECKER(One,I)           /  Kroneckerin tulo 2 ja
*MAT J=#HADAMARD(J1,J2)            /  näiden Hadamardin tulo,
*MAT LOAD J 111 CUR+1              /  valintamatriisi (16 x 4)
*MATRIX J
*Hadamard(KRONECKER(IDN,CON),KRONECKER(CON,IDN))
*///      1*1 2*1 3*1 4*1
*1*1       1   0   0   0
*1*2       0   0   0   0
*1*3       0   0   0   0
*1*4       0   0   0   0
*2*1       0   0   0   0
*2*2       0   1   0   0
*2*3       0   0   0   0
*2*4       0   0   0   0
*3*1       0   0   0   0
*3*2       0   0   0   0
*3*3       0   0   1   0
*3*4       0   0   0   0
*4*1       0   0   0   0
*4*2       0   0   0   0
*4*3       0   0   0   0
*4*4       0   0   0   1
```

Kertomalla edellä saatu matriisi $\boldsymbol{K}$ oikealta matriisilla $\boldsymbol{J}$ saadaan vihdoin ensi vaiheessa tavoiteltu yhdistelmä $\boldsymbol{Q}_{1A}$, joka näyttää jo tutummalta. Se paljastaa itse asiassa ristikon molemmat mahdolliset ratkaisut koodatussa muodossa:

```
*MAT Q1A=K*J
*MAT LOAD Q1A 111 CUR+1  /  kannattaa verrata matriisin sisäistä nimeä alla esitettyyn kaavaan!
*MATRIX Q1A
*(KRONECKER(P1,1*1')+KRONECKER(1*1',2*PA))*Hadamard(KRONECKER(IDN,CON),KRONECKER(CON,IDN))
*///      1*1 2*1 3*1 4*1
*1*1        3   1   0   2
*1*2        1   3   2   0
```

Matriisimerkinnöin nähdään, että kyseessä on **aika mielenkiintoinen lauseke**. Sen ytimessä on edelleen matriisien $\boldsymbol{P}_1$ ja $2\boldsymbol{P}_A$ summa, mutta ympärillä esiintyy peräti **kolme eri tyyppistä matriisituloa:** tavallinen skalaaritulo, Kroneckerin tulo (neljään kertaan!) ja Hadamardin tulo:

$$\boldsymbol{Q}_{1A} = \big[(\boldsymbol{P}_1 \otimes \mathbf{1}\mathbf{1}') + (\mathbf{1}\mathbf{1}' \otimes 2\boldsymbol{P}_A)\big]\big[(\boldsymbol{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \boldsymbol{I})\big].$$

Survon matriisitulkissa sarakkeiden poimimiseen on suoraviivaisempikin tapa. Sarakkeet osoitetaan binäärivektorilla, joka voidaan muodostaa kätevästi, kun havaitaan, että edellä mainitut "pääsarakkeet" vastaavat 4×4 -yksikkömatriisin $\boldsymbol{I}$ diagonaalialkioita. Tällöin tarvitsee vain muuntaa $\boldsymbol{I}$ vektoriksi ja transponoida se. Sarakkeiden erottaminen tapahtuu sen jälkeen matriisitulkin SUB-funktiolla:

```
*MAT H=VEC(I)'          /  vektoroidaan ja transponoidaan yksikkömatriisi
*MAT LOAD H 111 CUR+1
*MATRIX H
*VEC(IDN)'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*  1        1   0   0   0   0   1   0   0   0   0   1   0   0   0   0   1
*
*MAT Q1A=SUB(K,*,H)     /  poimitaan kaikki rivit ja H:n osoittamat sarakkeet
*MAT LOAD Q1A 111 CUR+1
*MATRIX Q1A
*SUB(KRONECKER(P1,1*1')+KRONECKER(1*1',2*PA),*,H)
*///      1*1 2*2 3*3 4*4
*1*1        3   1   0   2
*1*2        1   3   2   0
```

2.5.3 Koodaukset yhdistettynä

Kuten alussa esitetyssä esimerkissä, on tässäkin ratkaisun kannalta sinänsä turha jatkaa pidemmälle, koska ratkaisut näkyvät jo äsken saadusta matriisista $\boldsymbol{Q}_{1A}$. Se on kuitenkin havainnollisuuden vuoksi perusteltua, jotta nähdään miten koodausten yhdistäminen toimii yleisemmässä tilanteessa.

Aiemmin esitetyn perusteella saadaan samaan tapaan $\boldsymbol{P}_2$, $\boldsymbol{P}_B$ ja $\boldsymbol{Q}_{2B}$:

```
*MAT DIM P2 /* rowP2=1 colP2=4
*MAT DIM PB /* rowPB=2 colPB=4
*MAT One2!=CON(rowP2,1)        /  Sisäiset nimet tässä vähän
*MAT OneB!=CON(rowPB,1)       /    eri tavalla kuin edellä.
*MAT Ones!=CON(colP2,1)      /      Ei olennaista merkitystä,
*MAT Ones2=One2*Ones'       /        erot näkyvät matriisien
*MAT OnesB=OneB*Ones'      /          "historiankirjoituksessa"
*
*MAT Q2B=SUB(KRONECKER(P2,OnesB)+KRONECKER(Ones2,2*PB),*,H)  /  kaikki kerralla
*MAT LOAD Q2B 111 CUR+1
*MATRIX Q2B
*SUB(KRONECKER(P2,OneB*Ones')+KRONECKER(One2*Ones',2*PB),*,H)
*///      1*1 2*2 3*3 4*4
*1*1        0   2   3   1
*1*2        2   0   1   3
```

Koodataan sen jälkeen $\boldsymbol{Q}_{1A}$ ja $\boldsymbol{Q}_{2B}$ uudelleen edeltä tutuilla muunnoksilla:

```
*MAT TRANSFORM Q1A BY 2*X#+1  /  uudelleenkoodaus: {1,3,5,7}
*MAT TRANSFORM Q2B BY 2^X#    /  uudelleenkoodaus: {1,2,4,8}
*
*MAT LOAD Q1A 111 CUR+1
*MATRIX Q1A
*T(Q1A_by_2*X#+1)
*///      1*1 2*2 3*3 4*4
*1*1        7   3   1   5
*1*2        3   7   5   1
*
*MAT LOAD Q2B 111 CUR+1
*MATRIX Q2B
*T(Q2B_by_2^X#)
*///      1*1 2*2 3*3 4*4
*1*1        1   4   8   2
*1*2        4   1   2   8
```

Koska $\boldsymbol{Q}_{1A}$ ja $\boldsymbol{Q}_{2B}$ ovat tässä tapauksessa saman kokoisia, niiden yhdistämiseen voisi olla houkuttelevaa käyttää Hadamardin tuloa. Yleisesti se ei kuitenkaan riitä, koska pitää tarkastella **kaikkia mahdollisia rivien yhdistelmiä**. Niinpä sovelletaan Kroneckerin tuloa kuten edellä. Samalla nähdään myös, kuinka osa rivien yhdistelmistä on vääjäämättä ristikon ratkaisun kannalta kelpaamattomia.

```
*MAT Q1A2B=SUB(KRONECKER(Q1A,Q2B),*,H)    /  karsitaan turhat sarakkeet
*MAT DIM Q1A2B /* rowQ1A2B=4 colQ1A2B=4
*
*MAT CLABELS NUM(1) TO Q1A2B              /  sarakeotsikot  1,2,3,4 ja
*MAT RLABELS r      TO Q1A2B              /    riviotsikot r1,r2,r3,r4
*
*MAT LOAD Q1A2B 111 CUR+1
*MATRIX Q1A2B
*SUB(KRONECKER(T(Q1A_by_2*X#+1),T(Q2B_by_2^X#)),*,H)
*///        1   2   3   4
*r1         7  12   8  10
*r2        28   3   2  40
*r3         3  28  40   2
*r4        12   7  10   8
```

Matriisimerkinnöin koko tähänastinen yhdistely voidaan esittää muodossa

$$\boldsymbol{Q}_{1A2B} = (\boldsymbol{Q}_{1A} \otimes \boldsymbol{Q}_{2B})\boldsymbol{J},$$

jossa

$$\boldsymbol{Q}_{1A} = \big[(\boldsymbol{P}_1 \otimes \mathbf{1}\mathbf{1}') + (\mathbf{1}\mathbf{1}' \otimes 2\boldsymbol{P}_A)\big]\boldsymbol{J} \quad \text{ja} \quad \boldsymbol{Q}_{2B} = \big[(\boldsymbol{P}_2 \otimes \mathbf{1}\mathbf{1}') + (\mathbf{1}\mathbf{1}' \otimes 2\boldsymbol{P}_B)\big]\boldsymbol{J}$$

sekä $\boldsymbol{J} = (\boldsymbol{I} \otimes \mathbf{1}) \circ (\mathbf{1} \otimes \boldsymbol{I})$, mutta lyhin tapa ilmaista tämä on Kroneckerin tulon erikoismuoto, joka tunnetaan *Raon ja Khatrin tulona*[5]

$$\boldsymbol{Q}_{1A2B} = \boldsymbol{Q}_{1A} \odot \boldsymbol{Q}_{2B}.$$

Edellä on nähty, että 2×2 -ristikon ratkaisupohja sisältää koodit 7, 8, 10 ja 12, joten ratkaisun kannalta matriisin $\boldsymbol{Q}_{1A2B}$ riveistä kelvollisia ovat `r1` ja `r4`. Ne vastaavat täsmälleen ristikon kahta mahdollista ratkaisua. Edellinen on esitetty sivulla 1, jälkimmäisessä ristikon pystyrivit vain vaihtavat paikkaa (jolloin tietenkin kaikki luvut siirtyvät vastaavasti eri paikkoihin).

Rivit `r2` ja `r3` eivät kelpaa, koska yhdistelmät ovat **loogisesti mahdottomia**. Esimerkiksi koodi 28, joka on koodien 7 ja 4 tulo, tarkoittaisi sanallisesti mahdotonta yhdistelmää ”vaakarivin ja pystyrivin yhteinen luku” **ja** ”pystyrivin oma luku”.

[5]C. R. Rao 1920– ja C. G. Khatri 1931–1989, ks. `http://fi.wikipedia.org/wiki/C._R._Rao`

3 Suuren Survo-ristikon ratkaiseminen

On selvää, että edellä käsitelty 2×2 -kokoinen Survo-ristikko on pieni. Sen sijaan ei ole läheskään yhtä selvää, mitä tarkoittaa ”suuri Survo-ristikko”. Periaatteessa Survo-ristikon vaaka- ja pystyrivien lukumäärät m ja n voivat olla kuinka suuria tahansa, mutta käytännössä ne ovat aina pieniä lukuja. Yksi tyypillisimmistä tapauksista lienee $m = n = 4$. Jos $n > 4$, on yleensä $m \leq 4$, ja mitä suurempi n, sitä pienempi m. Esimerkiksi jos $n = 10$, on tyypillistä, että $m = 2$ tai $m = 3$.

Avointa 5×5 -ristikkoa voi hyvällä syyllä pitää jo varsin suurena haasteena. Toistaiseksi ei ole edes selvillä, montako sen kokoista, yksikäsitteisesti ratkeavaa Survo-ristikkoa on ylipäätään olemassa![6]

Suurempia kuin 5×5 -kokoisia neliöristikoita ei juuri ole esitetty. Harvinainen esimerkki on 6×6 -kokoinen Tehtävä 12 (Mustonen 2006a, 25), jonka vaikeusaste on peräti 17000 huolimatta siitä, että ristikon 36 luvusta 16 on annettu valmiiksi[7]. Survo-ristikon ”suuruus” viittaa siis pikemminkin eri ratkaisuvaiheissa kohdattavien vaihtoehtojen lukumääriin kuin ristikon ”fyysiseen” kokoon.

Tarkastellaan seuraavaksi, millä tavalla ”suuren” Survo-ristikon ratkaiseminen sujuu matriisien ja matriisioperaatioiden avulla. Siinäkin yhteydessä havaitaan, kuinka erilaisten vaihtoehtojen lukumäärät saattavat kasvaa varsin suuriksi.

3.1 Ratkaistava ristikko

Valitaan ratkaistavaksi Survo-ristikko 188/2010 (Mustonen 2010). Kyseessä on avoin 4×4 -ristikko, jonka vaikeusaste on 870. Alla näkyvät ristikko, vastaava koodausristikko (ks. kohta 2.4) sekä ristikon ratkaisu:

	A	B	C	D	
1					27
2					12
3					52
4					45
	20	45	31	40	

	A	B	C	D
1	7	12	33	3
2	10	8	22	2
3	45	36	13	9
4	5	4	11	1

	A	B	C	D	
1	3	10	6	8	27
2	1	5	2	4	12
3	9	16	12	15	52
4	7	14	11	13	45
	20	45	31	40	

Ratkaistaan nyt ristikko vaiheittain Survon eri toimintojen avulla. Osituksia luetellaan jälleen COMB-komennolla, mutta suoraan tekstitiedostoihin, joista luettelot siirretään matriiseiksi. Apuna tarvitaan Survon datatiedostoja, joita käsitellään erilaisilla FILE-operaatioilla. Muuttujamuunnoksia ja johdettuja muuttujia rakennellaan VAR-, TRANSFORM- ja VARSTAT-komennoilla.

Erikoisin komennoista on XALL, joka ei ole kuulunut Survon perusversioon, mutta on ollut aikoinaan (vuonna 1997) saatavilla Survon käyttäjäyhdistyksen lehden mukana olleella **SURVOTUT 4** -levykkeellä[8]. Tein XALL:in alun perin kyselyaineiston muokkausta varten. Sen uusiokäyttö 10 vuotta myöhemmin oli hauska yllätys!

[6] Aihepiiriä valottaa tarkemmin mm. Mustonen (2007). Ks. myös 18.4.2011 Survon keskustelupalstalla käyty viestinvaihto Matti K. Sinisalon ja Seppo Mustosen välillä: näyttää siltä, että eräänlainen *alaraja* ao. lukumäärälle olisi n. 44 miljoonaa.

[7] Onnistuin ratkaisemaan tämän ”petomaisen” ristikon Survon COMB-komennon ja editoriaalisen laskennan avulla 30.5.–1.6.2006 tutustuessani Mustosen artikkelin (2006a) alustavaan versioon. Laskelmani ja kokeiluni käsittävät lähes 5500 riviä yhdessä Survon toimituskentässä.

[8] **SURVO.TUT**-nimisen lehden ja **SURVOTUT**-levykkeiden sisältöihin voi tutustua sivulla `http://www.survo.fi/yhdistys/survotut.html`.

3.2 Ositukset binäärimuodossa ja koodattuna

Ratkaisu tapahtuu lopulta automaattisesti yhdellä aktivoinnilla, kun kaikki komentokaaviot on ensin rakennettu valmiiksi. Tässä kaavioita tarkastellaan paloittain, mutta todellisuudessa ne ovat peräkkäin yhdessä toimituskentässä.

```
*/ACTIVATE +  /  tämän aktivointi aktivoi kaikki rivit, joiden kontrollimerkkinä on +
*---- Vaakarivi 1: --------------------
+COMB P1 TO P1.TXT / P1=PARTITIONS,27,4  MIN=1 MAX=16 DISTINCT=1
+FILE MAKE X1,16,0,X,1    /   Luodaan tyhjä datatiedosto muuttujinaan X1,X2,...,X16
+FILE SAVE P1.TXT TO X1   /   Viedään ositukset tekstitiedostosta datatiedostoon
+XALL X1,X1,16            /   Siirretään tiedot oikeille paikoilleen
+TRANSFORM X1 BY NARY     /   Muunnetaan data BInäärimuotoon: NARY=if(X=MISSING)then(0)else(1)
+MAT SAVE DATA X1 TO P1   /   Talletetaan datatiedosto binäärimatriisiksi
*---- Pystyrivi A: --------------------
+COMB PA TO PA.TXT / PA=PARTITIONS,20,4
+FILE MAKE XA,16,0,X,1
+FILE SAVE PA.TXT TO XA
+XALL XA,X1,16
+TRANSFORM XA BY NARY
+MAT SAVE DATA XA TO PA
*---- Tarvittavat ykkösmatriisit: -----
+MAT DIM P1 /* rowP1=61 colP1=16
+MAT DIM PA /* rowPA=23 colPA=16
+MAT One1!=CON(rowP1,1)
+MAT OneA!=CON(rowPA,1)
+MAT Ones!=CON(colP1,1)
+MAT Ones1=One1*Ones'         / *Ones1~One1*Ones' 61*16
+MAT OnesA=OneA*Ones'         / *OnesA~OneA*Ones' 23*16
*
*---- Vaakarivi 2 ja pystyrivi B: -----
+COMB P2 TO P2.TXT / P2=PARTITIONS,12,4
+FILE MAKE X2,16,0,X,1
+FILE SAVE P2.TXT TO X2
+XALL X2,X1,16
+TRANSFORM X2 BY NARY
+MAT SAVE DATA X2 TO P2
*
+COMB PB TO PB.TXT / PB=PARTITIONS,45,4
+FILE MAKE XB,16,0,X,1
+FILE SAVE PB.TXT TO XB
+XALL XB,X1,16
+TRANSFORM XB BY NARY
+MAT SAVE DATA XB TO PB
*
+MAT DIM P2 /* rowP2=2 colP2=16
+MAT DIM PB /* rowPB=38 colPB=16
+MAT One2!=CON(rowP2,1)
+MAT OneB!=CON(rowPB,1)
+MAT Ones!=CON(colP2,1)
+MAT Ones2=One2*Ones'         / *Ones2~One2*Ones' 2*16
+MAT OnesB=OneB*Ones'         / *OnesB~OneB*Ones' 38*16
*
*---- Vaakarivi 3 ja pystyrivi C: -----
+COMB P3 TO P3.TXT / P3=PARTITIONS,52,4
+FILE MAKE X3,16,0,X,1
+FILE SAVE P3.TXT TO X3
+XALL X3,X1,16
+TRANSFORM X3 BY NARY
+MAT SAVE DATA X3 TO P3
*
+COMB PC TO PC.TXT / PC=PARTITIONS,31,4
+FILE MAKE XC,16,0,X,1
+FILE SAVE PC.TXT TO XC
+XALL XC,X1,16
+TRANSFORM XC BY NARY
+MAT SAVE DATA XC TO PC
*
+MAT DIM P3 /* rowP3=9 colP3=16
+MAT DIM PC /* rowPC=79 colPC=16
+MAT One3!=CON(rowP3,1)
+MAT OneC!=CON(rowPC,1)
+MAT Ones!=CON(colP3,1)
+MAT Ones3=One3*Ones'         / *Ones3~One3*Ones' 9*16
+MAT OnesC=OneC*Ones'         / *OnesC~OneC*Ones' 79*16
```

Sivun 12 kaavioilla saadaan siis aikaan binääriset versiot kolmen ensimmäisen vaaka- ja pystyriviparin mahdollisista osituksista, siis matriisit $\boldsymbol{P}_1$ ja $\boldsymbol{P}_A$, $\boldsymbol{P}_2$ ja $\boldsymbol{P}_B$ sekä $\boldsymbol{P}_3$ ja $\boldsymbol{P}_C$. Kuten matriisien dimensioista nähdään, ovat ositusten lukumäärät vastaavasti 61, 23, 2, 38, 9 ja 79.

Lisäksi on luotu valmiiksi em. matriisien yhdistelyssä tarvittavat ykkösmatriisit. Yhdistelyt matriiseiksi $\boldsymbol{P}_{1A}$, $\boldsymbol{P}_{2B}$ ja $\boldsymbol{P}_{3C}$ tapahtuvat Raon ja Khatrin tuloina:

```
+MAT H=VEC(IDN(16,16))'  /  16 x 16 -yksikkömatriisista 1 x 256 -valintavektori
+MAT P1A=SUB(KRONECKER(P1,OnesA)+KRONECKER(Ones1,2*PA),*,H)
+MAT P2B=SUB(KRONECKER(P2,OnesB)+KRONECKER(Ones2,2*PB),*,H)
+MAT P3C=SUB(KRONECKER(P3,OnesC)+KRONECKER(Ones3,2*PC),*,H)
```

Ositusten lukumäärät vaikuttavat kohtuullisilta, mutta niiden yhdistelmien lukumäärä kasvaa äkkiä todella suureksi. Kaikki yhdistelmät eivät kuitenkaan kelpaa, kuten aiemmin nähtiin. Jotta matriisien dimensiot eivät kasvaisi liian suuriksi, on epäkelvot yhdistelmät karsittava pois. Tätä on tehtävä useaan kertaan, sillä jokainen Kroneckerin tulo tuottaa kaikki yhdistelmät, olivat ne kelpoja tai ei.

Karsitaan siis nyt myös matriisien rivejä. Alkutilanteessa äsken muodostetuissa kolmessa matriisissa on $61 \cdot 23 = 1403$, $2 \cdot 38 = 76$ ja $9 \cdot 79 = 711$ riviä. Näistä kelpaavat vain ne, joissa on kolme ykköstä, kolme kakkosta ja yksi kolmonen, jolloin loput yhdeksän koodia ovat nollia (vrt. kohta 2.4). Karsinta tapahtuu siirtämällä matriisit datatiedostoiksi, laskemalla koodien lukumäärät riveittäin ja tallettamalla takaisin matriiseiksi vain ne rivit, jotka täyttävät mainitut kelpoisuusehdot:

```
+MAT DIM P1A /* rowP1A=1403 colP1A=16
+MAT DIM P2B /* rowP2B=76 colP2B=16
+MAT DIM P3C /* rowP3C=711 colP3C=16
+FILE DEL P1A                              /  poistetaan mahdolliset
+FILE DEL P2B                              /   aiemmat datatiedostot
+FILE DEL P3C                              /
+FILE SAVE MAT P1A TO P1A / TYPE=1         /  matriisit datoiksi, muuttujat
+FILE SAVE MAT P2B TO P2B                  /   minimikokoa (1 tavu)
+FILE SAVE MAT P3C TO P3C                  /
+FILE MASK P1A,CASE,1,-                    /  havainnon tunnistemuuttuja
+FILE MASK P2B,CASE,1,-                    /   CASE (matriisin riviotsikko)
+FILE MASK P3C,CASE,1,-                    /   passiiviseksi ennen laskentaa
+VARSTAT P1A / VARSTAT=N1:1,N2:1,N3:1      /
+VARSTAT P2B / N1=#VAL,1 N2=#VAL,2         /  lasketaan koodien lukumäärät
+VARSTAT P3C / N3=#VAL,3                   /
*.................................       /  rajarivi
+FILE MASK P1A,CASE,1,A                    /
+FILE MASK P2B,CASE,1,A                    /  CASE takaisin aktiiviseksi
+FILE MASK P3C,CASE,1,A                    /
+FILE MASK P1A,N1,1,-...                   /  koodien lukumäärämuuttujat
+FILE MASK P2B,N1,1,-...                   /   passiivisiksi
+FILE MASK P3C,N1,1,-...                   /
+MAT SAVE DATA P1A TO Q1A / SELECT=A*B*C   /  datat takaisin matriiseiksi
+MAT SAVE DATA P2B TO Q2B / A=N1,3 B=N2,3  /   niiltä osin kuin havainnot
+MAT SAVE DATA P3C TO Q3C / C=N3,1         /   täyttävät kelpoisuusehdot
+MAT DIM Q1A /* rowQ1A=695 colQ1A=16
+MAT DIM Q2B /* rowQ2B=25 colQ2B=16
+MAT DIM Q3C /* rowQ3C=405 colQ3C=16
*.................................
```

Karsinnan tuloksena muodostuvat uudet matriisit $\boldsymbol{Q}_{1A}$, $\boldsymbol{Q}_{2B}$ ja $\boldsymbol{Q}_{3C}$, joiden rivien lukumääriksi saadaan 695, 25 ja 405. Ero on merkittävä, sillä ilman karsintaa näiden matriisien yhdistely Raon ja Khatrin tulon avulla johtaisi matriisiin, jossa olisi $1403 \cdot 76 \cdot 711 = 75812508$ *(yli 75 miljoonaa)* riviä. Sen sarakkeiden lukumäärä olisi (karsinnan jälkeen) 16, jolloin matriisissa olisi **yli miljardi** alkiota. Survon matriisitulkki tallettaa alkiot kaksoistarkkuudella, jossa jokainen alkio vie 8 tavua tilaa. Karsinta kannattaa, sillä kyseisen matriisin käsittely vaatisi jo lähes **10 gigatavua** keskusmuistia!

3.3 Koodaukset yhdistettynä

Suoritetaan sitten karsittujen matriisien uudelleenkoodaus jo tuttuun tapaan ja yhdistetään sen jälkeen matriisit $\boldsymbol{Q}_{1A}$ ja $\boldsymbol{Q}_{2B}$ Raon ja Khatrin tulon avulla:

```
+MAT #TRANSFORM Q1A BY U  /  {0,1,2,3} -> {1,3,5,7} U=2*X#+1
+MAT #TRANSFORM Q2B BY V  /  {0,1,2,3} -> {1,2,4,8}  V=2^X#
+MAT #TRANSFORM Q3C BY W  /  {0,1,2,3} -> {1,9,11,13} W=if(X#=0)then(1)else(a)
*   a=if(X#=1)then(9)else(b) b=if(X#=2)then(11)else(c) c=if(X#=3)then(13)else(X#)
*
+MAT Q1A2B=#RAO_KHATRI(Q1A,Q2B) / tässä hyödynnetään uutta matriisioperaatiota (ks. kohta 4.4)
+MAT DIM Q1A2B /* rowQ1A2B=17375 colQ1A2B=16
```

Yhdistelmiä on $695 \cdot 25 = 17375$, joista luultavasti suurin osa on epäkelpoja. Niiden karsinta tapahtuu vastaavasti kuin edellä; ehtoja vain on enemmän. Nyt ollaan jo tilanteessa, jota vastaava koodausmatriisi on esitetty sivulla 4.

```
+FILE DEL Q1A2B
+FILE SAVE MAT Q1A2B TO Q1A2B / TYPE=1
+FILE MASK Q1A2B,CASE,1,-
+FILE EXPAND Q1A2B  /  lisätään tilaa uusille muuttujille
*
+VARSTAT Q1A2B / VARSTAT=N1:1,N2:1,N3:1,N4:1,N5:1,N7:1,N8:1,N10:1,N12:1
*  N1=#VAL,1  N2=#VAL,2  N3=#VAL,3    N4=#VAL,4    N5=#VAL,5
*  N7=#VAL,7  N8=#VAL,8  N10=#VAL,10  N12=#VAL,12
*.......................
+FILE MASK Q1A2B,CASE,1,A
+FILE MASK Q1A2B,N1,1,-...
+MAT SAVE DATA Q1A2B TO Q1A2B / SELECT=A*B*C*D*E*F*G*H*I
*  A=N1,4 B=N2,2 C=N3,2 D=N4,2 E=N5,2 F=N7,1 G=N8,1 H=N10,1 I=N12,1
+MAT DIM Q1A2B /* rowQ1A2B=3 colQ1A2B=16
*.......................
```

Karsinta osoittautuu tällä kohtaa erityisen tehokkaaksi, sillä vain **kolme yhdistelmää** kelpaa jatkoon. Viimeisen yhdistelyn lopputuloksena saadaan matriisi $\boldsymbol{Q}_{1A2B3C}$, jossa on aluksi $3 \cdot 405 = 1215$ riviä:

```
+MAT Q1A2B3C=#RAO_KHATRI(Q1A2B,Q3C)
+MAT DIM Q1A2B3C /* rowQ1A2B3C=1215 colQ1A2B3C=16
```

Viimeiseen karsintaan liittyvä koodausmatriisi on esitetty kohdassa 3.1 sivulla 11. Kun jokainen koodi on yksikäsitteinen, jäljelle jää ainoastaan yksi yhdistelmä:

```
+FILE DEL Q1A2B3C
+FILE SAVE MAT Q1A2B3C TO Q1A2B3C / TYPE=1
+FILE MASK Q1A2B3C,CASE,1,-
+FILE EXPAND Q1A2B3C
*
+VARSTAT Q1A2B3C / VARSTAT=N1:1,N2:1,N3:1,N4:1,N5:1,N7:1,N8:1,N10:1,N12:1,&
*                          N9:1,N11:1,N13:1,N22:1,N33:1,N36:1,N45:1
*  N1=#VAL,1    N2=#VAL,2    N3=#VAL,3    N4=#VAL,4    N5=#VAL,5
*  N7=#VAL,7    N8=#VAL,8    N10=#VAL,10  N12=#VAL,12
*  N9=#VAL,9    N11=#VAL,11  N13=#VAL,13  N22=#VAL,22
*  N33=#VAL,33  N36=#VAL,36  N45=#VAL,45
*.......................
+FILE MASK Q1A2B3C,CASE,1,A
+FILE MASK Q1A2B3C,N1,1,-...
+MAT SAVE DATA Q1A2B3C TO Q1A2B3C / SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P
*  A=N1,1  B=N2,1 C=N3,1  D=N4,1  E=N5,1  F=N7,1  G=N8,1  H=N10,1
*  I=N12,1 J=N9,1 K=N11,1 L=N13,1 M=N22,1 N=N33,1 O=N36,1 P=N45,1
+MAT DIM Q1A2B3C /* rowQ1A2B3C=1 colQ1A2B3C=16
```

Ratkaisun avaimet ovat käsillä, joten on enää varmistettava niiden toimivuus. Senkin voi tehdä havainnollisesti Survolla, kuten seuraavassa nähdään.

3.4 Loppuhuipennus: luvut paikoilleen

Ratkaisun loppuhuipennuksena sijoitellaan luvut 1–16 oikeille paikoilleen:

```
+LOADM X1A2B3C 111 CUR+1          /  kohdassa 2.4 rakennettu koodausmatriisi
*4x4
*           A   B   C   D
*  1        7  12  33   3
*  2       10   8  22   2
*  3       45  36  13   9
*  4        5   4  11   1
*
*Muunnetaan aluksi koodausmatriisi vektoriksi:
+MAT V1A2B3C=VEC(X1A2B3C)          /  neliömatriisi pystyvektoriksi
+MAT V1A2B3C(0,1)="koodi"          /  sarakeotsikko
+MAT LOAD V1A2B3C' 111 CUR+1       /  vilkaistaan vaakamuodossa
*MATRIX V1A2B3C'
*VEC(4x4)'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*koodi      7  10  45   5  12   8  36   4  33  22  13  11   3   2   9   1
*
*Muokataan ratkaisuvektori vastaavaan muotoon:
+MAT CLABELS NUM(1) TO Q1A2B3C     /  sarakeotsikot 1,2,3,...,16
+MAT Q1A2B3C(1,0)="koodi"          /  riviotsikko
+MAT P1A2B3C=Q1A2B3C'              /  muunnetaan pystyvektoriksi
+MAT LOAD P1A2B3C' 111 CUR+1       /  vilkaistaan vaakamuodossa
*MATRIX P1A2B3C'
*///        1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*koodi     10  22   7   2   8  33   5   3  45  12  11  13   1   4   9  36
*
*Talletetaan vektorit datoiksi; muuttujaan CASE tulee luvut 1,2,3,...,16:
+FILE DEL KOODAUS
+FILE DEL RATKAISU
+FILE SAVE MAT V1A2B3C TO KOODAUS  /  TYPE=1
+FILE SAVE MAT P1A2B3C TO RATKAISU
*..........................
*
*Täsmäytetään datat hakemalla koodeja vastaavat varsinaiset luvut:
*
+FILE COPY RATKAISU TO KOODAUS     /  VARS=CASE MATCH=koodi MODE=2
*
*Luvut 1,2,3,...,16 ovat siis molemmissa datoissa muuttujana CASE.
*KOODAUS-datassa kutakin niistä vastaa yksikäsitteinen paikkakoodi
*muuttujana koodi (1,2,3,4,5,7,8,10,11,12,...,45). Esimerkiksi sen
*koodilla 36 on CASE-arvo 7, ts. koodi 36 vastaa ristikon paikkaa B3
*(siis täsmälleen 7. alkio, kun matriisia tarkastellaan vektorina).
*
*RATKAISU-datassa luvut 1,2,3,...,16 tarkoittavat ristikon ratkaisua,
*ja koodi kertoo vastaavasti, mihin kohtaan kukin luvuista sijoittuu.
*FILE COPY -komento siirtää nyt "ratkaisun avaimet" KOODAUS-dataan, ja
*täsmennys MATCH määrää, mille kohtaa ne (CASE-luvut) sijoitetaan.
*(MODE-täsmennyksen ansiosta datoja ei tarvitse mitenkään järjestää.)
*Kaaviona esitettynä saadaan siis selville, mihin tulee mitäkin:
*
*CASE:      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
*koodi:     7  10  45   5  12   8  36   4  33  22  13  11   3   2   9   1
*luvut:     3   1   9   7  10   5  16  14   6   2  12  11   8   4  15  13
*.....................................................................
*
+VAR koodi=CASE TO KOODAUS         /  täsmäytetyt luvut koodien paikalle
+MAT SAVE DATA KOODAUS TO KOODAUS  /  datan siirto 16 x 1 -matriisiksi
+MAT KOODAUS=VEC(KOODAUS,4)        /  pystyvektorista neliömatriisiksi
+MAT RLABELS FROM X TO KOODAUS     /  rivi- ja sarakeotsikot lopulta
+MAT CLABELS FROM X TO KOODAUS     /   kuten alun perin kohdassa 2.4
+MAT NAME KOODAUS AS Ratkaisu      /  koodauksella aikaansaatu ratkaisu!
*
+LOADM KOODAUS 111 CUR+2 / SUMS=1
*
*Ratkaisu
*           A   B   C   D  Sum
*  1        3  10   6   8  27
*  2        1   5   2   4  12
*  3        9  16  12  15  52
*  4        7  14  11  13  45
*Sum       20  45  31  40
```

4 Täydennyksiä ja huomautuksia

Lopuksi esitän pieniä täydennyksiä ja huomautuksia koskien suorakaideristikoita, koodien valintaa, valmiiksi annettuja lukuja sekä eräitä uusia matriisioperaatioita.

4.1 Suorakaideristikot

Vaikka heti sivulla 1 kerrotaan, että Survo-ristikon dimensiot m ja n voivat olla eri lukuja, on tähänastisissa esimerkeissä keskitytty vain tapauksiin, joissa $m = n$. Ratkaisutapani soveltuukin parhaiten näihin *neliöristikoihin*, kun ratkaisu etenee aina yksi vaakarivi ja pystyrivi kerrallaan.

Olin kuitenkin jo alun perin intuitiivisesti sitä mieltä, että ratkaisutapani toimii myös *suorakaideristikoilla* (joissa siis $m \neq n$), mutten ollut syventynyt asiaan ennen tämän paperin valmistelua. Intuitioni osoittautui lopulta oikeaksi. Suorakaideristikoiden ratkaiseminen matriiseilla perustuu ”rivien kierrätykseen”: samoja rivejä hyödynnetään useaan kertaan. Voisi puhua myös ”haamuriveistä”.

Totean ratkaisutapani toimivuuden suorakaideristikoilla esimerkillä, jossa tarkastellaan helpon 2×5 -ristikon ratkaisemista. Ensin muodostetaan vastaava koodausristikko ja sen jälkeen ratkaistaan Survo-ristikko 12/2006 (Mustonen 2006b). Suuri osa komentokaavioista on aivan vastaavia kuin kohdassa 3.2 (s. 12–13), joten niitä ei tässä toisteta. Ristikon helppoudesta (vaikeusaste vain 15) johtuen turhat vaihtoehdot on karsittu ”käsin” tutkailemalla tulosmatriiseja toimituskentässä.

```
*/ACTIVATE +     /  aktivoidaan taas kaikki rivit, joiden kontrollisarakkeessa on +
*
+MAT X=ZER(2,5)  /  Muodostetaan 2 x 5 -nollamatriisi X ja
+MAT X(0,1)="A"  /  otsikoidaan se Survo-ristikon tapaan.
+MAT X(0,2)="B"  /  Haamurivejä lukuunottamatta tässä edetään
+MAT X(0,3)="C"  /  aivan vastaavasti kuin kohdassa 2.4.
+MAT X(0,4)="D"
+MAT X(0,5)="E"
+MAT RLABELS NUM(1) TO X
+MAT X1=X
+MAT XA=X
+MAT X2=X
+MAT XB=X
+MAT X3=X
+MAT XC=X
+MAT X4=X
+MAT XD=X
*
+MAT #TRANSFORM X1 BY F1 / F1=if(I#=1)then(1)else(0)
+MAT #TRANSFORM XA BY FA / FA=if(J#=1)then(1)else(0)
+MAT #TRANSFORM X2 BY F2 / F2=if(I#=2)then(1)else(0)
+MAT #TRANSFORM XB BY FB / FB=if(J#=2)then(1)else(0)
+MAT #TRANSFORM X3 BY F1 /                              "haamurivi 3" = rivi 1
+MAT #TRANSFORM XC BY FC / FC=if(J#=3)then(1)else(0)
+MAT #TRANSFORM X4 BY F2 /                              "haamurivi 4" = rivi 2
+MAT #TRANSFORM XD BY FD / FD=if(J#=4)then(1)else(0)
*
+MAT X1A=X1+2*XA
+MAT X2B=X2+2*XB
+MAT X3C=X3+2*XC
+MAT X4D=X4+2*XD
*                          U=2*X#+1                    V=2^X#
+MAT #TRANSFORM X1A BY U / W=if(X#=0)then(01)else(a)  Z=if(X#=0)then(01)else(d)
+MAT #TRANSFORM X2B BY V / a=if(X#=1)then(09)else(b)  d=if(X#=1)then(16)else(e)
+MAT #TRANSFORM X3C BY W / b=if(X#=2)then(11)else(c)  e=if(X#=2)then(17)else(f)
+MAT #TRANSFORM X4D BY Z / c=if(X#=3)then(13)else(X#) f=if(X#=3)then(19)else(X#)
*
+MAT X1A2B=#HADAMARD(X1A,X2B)
+MAT X1A2B3C=#HADAMARD(X1A2B,X3C)
+MAT X1A2B3C4D=#HADAMARD(X1A2B3C,X4D)
*
```

Otetaan näin muodostettu koodausristikko näkyville sen eri vaiheissa:

```
+LOADM X1A2B     111 CUR+1 / 2 x 5 -koodausristikko, vaihe 1
*Hadamard(T(X1A_by_U),T(X2B_by_V))
*           A   B   C   D   E
*  1        7  12   3   3   3
*  2       10   8   2   2   2
*
+LOADM X1A2B3C   111 CUR+1 / 2 x 5 -koodausristikko, vaihe 2
*Hadamard(Hadamard(T(X1A_by_U),T(X2B_by_V)),T(X3C_by_W))
*           A   B   C   D   E
*  1       63 108  39  27  27
*  2       10   8  22   2   2
*
+LOADM X1A2B3C4D 111 CUR+1 / 2 x 5 -koodausristikko, vaihe 3
*Hadamard(Hadamard(Hadamard(T(X1A_by_U),T(X2B_by_V)),T(X3C_by_W)),T(X4D_by_Z))
*           A   B   C   D   E
*  1       63 108  39 459  27
*  2      160 128 352  38  32
```

Ryhdytään sitten ratkaisemaan valittua ristikkoa:

```
*Survo-ristikko 12/2006
*    A  B  C  D  E         Vaikeusaste on vain 15. Se sattuu olemaan myös rivin 1
*1   *  *  *  *  *  15     summa, joka on 1+2+3+4+5=15 (jossakin järjestyksessä).
*2   *  *  *  *  *  40     (Vastaavasti rivin 2 summa on tietenkin 6+7+8+9+10=40.)
*   13 15 11  7  9
*
*[...] (rakennetaan matriisit P1, PA, P2 ja PB sekä Q1A ja Q2B kuten kohdassa 3.2)
*MAT Q1A2B=#RAO_KHATRI(Q1A,Q2B)
*MAT LOAD Q1A2B 111 CUR+1 / (12 vaihtoehtoa)
*MATRIX Q1A2B
*Rao_Khatri(T(Q1A_by_2*X#+1),T(Q2B_by_2^X#))
*///      1*1 2*2 3*3 4*4 5*5 6*6 7*7 8*8 9*9 10*
*1*1*1*1    3   3   7   3  12   2   2   2   2  40
*1*1*1*2    3   3   7   3   3   8   2   2   8  10
*1*1*1*3    3   3   7   3   3   2   8   8   2  10
*1*2*1*1    3   3   3   7  12   2   2   2  10   8  <- ainoa kelvollinen
*1*2*1*2    3   3   3   7   3   8   2   2  40   2
*[...] (rivejä poistettu tilan säästämiseksi)
*
*Ratkaisua, vaihe 1:                                    Koodausristikko, vaihe 1:
*    A   B  C  D  E                                             A   B   C   D   E
*1   4   5  {1,2,3}  15    Neljä lukua siis kiinnitetty    1        7  12   3   3   3
*2   9  10  {6,7,8}  40    tässä vaiheessa, kuusi auki.    2       10   8   2   2   2
*   13  15 11  7  9
*
*MAT Q1A2B=Q1A2B(4,*)   /  vain kelvollinen jatkoon, muut karsitaan
```

Kun aidot vaakarivit 1 ja 2 on käytetty, siirrytään "haamuriveihin": kierrätetään riviä 1 ja muodostetaan matriisit $\boldsymbol{P}_C$ ja $\boldsymbol{Q}_{1C}$. Jälkimmäinen muunnetaan muunnoksella W: {1,9,11,13}, minkä jälkeen se yhdistetään aiempiin:

```
*MAT Q1A2B1C=#RAO_KHATRI(Q1A2B,Q1C)
*MAT LOAD Q1A2B1C 111 CUR+1 / (5 vaihtoehtoa)
*MATRIX Q1A2B1C
*Rao_Khatri(Q1A2B(4,*),T(Q1C_by_W))
*///      1*1 2*2 3*3 4*4 5*5 6*6 7*7 8*8 9*9 10*
*1*2*1*1*  39  27  27  63 108   2   2   2  10  88
*1*2*1*1*  27  39  27  63 108   2   2   2 110   8
*1*2*1*1*  27  27  39  63 108   2   2  22  10   8  <- ainoa kelvollinen
*1*2*1*1*  27  27  27  91 108   2  22   2  10   8
*1*2*1*1*  27  27  27  63 156  22   2   2  10   8
*
*Ratkaisua, vaihe 2:                                    Koodausristikko, vaihe 2:
*    A   B  C  D  E                                             A   B   C   D   E
*1   4   5  3  {1,2} 15    Kuusi lukua siis kiinnitetty    1       63 108  39  27  27
*2   9  10  8  {6,7} 40    tässä vaiheessa, neljä auki.    2       10   8  22   2   2
*   13  15 11  7  9
*
*MAT Q1A2B1C=Q1A2B1C(3,*)  /  vain kelvollinen jatkoon, muut karsitaan
```

Samalla tavalla vaakariviä 2 käytetään haamurivinä pystyrivin D kanssa, muodostetaan matriisit $\boldsymbol{P}_D$ ja $\boldsymbol{Q}_{2D}$, muunnetaan jälkimmäinen muunnoksella Z: {1,16,17,19} ja yhdistetään taas aiempiin. Ristikko ratkeaa:

```
*MAT Q1A2B1C2D=#RAO_KHATRI(Q1A2B1C,Q2D)
*MAT LOAD Q1A2B1C2D 1111 CUR+1 / (3 vaihtoehtoa)
*MATRIX Q1A2B1C2D
*Rao_Khatri(Q1A2B1C(3,*),T(Q2D_by_Z))
*///       1*1  2*2  3*3  4*4  5*5  6*6  7*7  8*8  9*9 10*1
*1*2*1*1*  459   27   39   63  108   38   32  352  160  128  <- ainoa kelvollinen
*1*2*1*1*   27  459   39   63 1836   32   32  352  160  128
*1*2*1*1*   27   27  663 1071  108   32   32  352  160  128
*
*Ratkaisu, vaihe 3:                                      Koodausristikko, vaihe 3:
*    A   B  C  D  E                                              A   B   C   D   E
*1   4   5  3  1  2  15    Kaikki luvut kiinnitetty        1       63 108  39 459  27
*2   9  10  8  6  7  40    yksikäsitteisesti - ratkaisu!   2      160 128 352  38  32
*   13  15 11  7  9
```

Ratkaisutapa siis toimii, mutta koodaus ei ole yhtä selkeää kuin neliöristikoissa.

4.2 Koodien valinta

Ratkaisutavassani keskeisessä asemassa ovat koodit, joita yhdistellään vaiheittain kertomalla niitä keskenään. Koodien valinnassa on olennaista saavuttaa yksikäsitteisyys: lopullisessa ratkaisussa erilaisia koodeja on oltava täsmälleen mn kpl, siis saman verran kuin ristikkoon tulevia lukuja. Koodit eivät saa missään vaiheessa sekoittua aiempiin koodeihin. Poikkeus on luku yksi, joka yksikäsitteistyy aina vasta lopullisessa (neliöristikon) ratkaisussa.

Ilmeisesti yksi tapa olisi valita muiksi koodeiksi pelkkiä **alkulukuja** (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...), mutta huolellisesti valitsemalla voidaan hyödyntää eräitä muitakin lukuja ja välttää turhan suurilla luvuilla operointi.

Alkuperäinen ideani lähti lukujen $x = \{0, 1, 2, 3\}$ uudelleenkoodauksesta sopivilla muunnoksilla. Huomasin, että muunnoksilla $2x+1$ ja 2^x saan kätevästi aikaan koodit $\{1, 3, 5, 7\}$ ja $\{1, 2, 4, 8\}$. Siitä eteenpäin en keksinyt yhtä kompakteja tapoja, mutta lopulta sain hahmotettua kelvolliset minimikoodit muutamaa seuraavaa vaihetta varten. Olennaista on ymmärtää, millä periaatteella koodeja yhdistellään.

Oheinen Survon editoriaalisen laskennan työkaavio näyttää vaiheiden 1–5 koodit ja joukon niiden yhdistelmiä:

```
*     koodit:      yhdistelmiä:
* 1:  1  3  5  7
* 2:  1  2  4  8    2*5=10   3*4=12
* 3:  1  9 11 13   2*11=22  3*11=33   4*9=36   5*9=45
* 4:  1 16 17 19   2*17=34  3*17=51  4*16=64  5*16=80   9*17=153  11*16=176
* 5:  1 21 23 25   2*23=46  3*23=69  4*21=84  5*21=105  9*23=207  11*21=231  17*21=357  16*23=368
```

Tarvittavat uudet yhdistelmät muodostuvat siis kertomalla ratkaistavan ristikon vaakarivin ja pystyrivin omia lukuja vastaavia keskimmäisiä koodeja (3 ja 5, 2 ja 4 jne.) *ristiin eri vaiheiden välillä*. Reunimmaiset koodit (1, 7, 8 jne.) eivät ykkösen ansiosta muutu miksikään.

Koska ratkaisutapani edetessä kertolaskut tapahtuvat Kroneckerin tuloina ja sen muunnelmina, ei tällaisia ristiin kertomisia voida erikseen huomioida, vaan kaikkia koodeja kerrotaan keskenään. Siitä syntyy myös turhia (epäloogisia) yhdistelmiä, jotka täytyy voida tunnistaa. Niinpä mitä tahansa lukuja ei voida hyödyntää koodeina. Esimerkiksi luvut 6 ($3 \cdot 2$) ja 14 ($2 \cdot 7$) ovat tällaisia. Luvut 15 ja 18 eivät puolestaan kelpaa sen vuoksi, että $3 \cdot 15 = 5 \cdot 9$ ja $2 \cdot 18 = 4 \cdot 9$.

Seuraavassa nähdään vaiheiden 1–5 koodit ja niiden yhdistelmät, kun rakennetaan 6×6 -koodausristikko. Alkuvalmistelut on jätetty tästä pois, koska ne ovat täysin samanlaiset kuin edellä.

```
*[...] (alkuvaiheet kuten kohdissa 2.4 ja 4.1)
+MAT #TRANSFORM X1B BY U  /  {0,1,2,3} -> {1,3,5,7}   U=2*X#+1
+MAT #TRANSFORM X2C BY V  /  {0,1,2,3} -> {1,2,4,8}   V=2^X#
+MAT #TRANSFORM X3E BY W  /  {0,1,2,3} -> {1,9,11,13}
+MAT #TRANSFORM X4F BY Z  /  {0,1,2,3} -> {1,16,17,19}
+MAT #TRANSFORM X5A BY Y  /  {0,1,2,3} -> {1,21,23,25}
*
*W=if(X#=0)then(01)else(a)  Z=if(X#=0)then(01)else(d)  Y=if(X#=0)then(01)else(g)
*a=if(X#=1)then(09)else(b)  d=if(X#=1)then(16)else(e)  g=if(X#=1)then(21)else(h)
*b=if(X#=2)then(11)else(c)  e=if(X#=2)then(17)else(f)  h=if(X#=2)then(23)else(i)
*c=if(X#=3)then(13)else(X#) f=if(X#=3)then(19)else(X#) i=if(X#=3)then(25)else(X#)
*
+MAT X1A2B=#HADAMARD(X1A,X2B)
+MAT X1A2B3C=#HADAMARD(X1A2B,X3C)
+MAT X1A2B3C4D=#HADAMARD(X1A2B3C,X4D)
+MAT X1A2B3C4D5E=#HADAMARD(X1A2B3C4D,X5E)
*
+MAT NAME X1A2B       AS 1A2B:
+MAT NAME X1A2B3C     AS 1A2B3C:
+MAT NAME X1A2B3C4D   AS 1A2B3C4D:
+MAT NAME X1A2B3C4D5E AS 1A2B3C4D5E:
*
+CLEAR CUR+8,END
+LOADM X1A2B       111 END+2
+LOADM X1A2B3C     111 END+2
+LOADM X1A2B3C4D   111 END+2
+LOADM X1A2B3C4D5E 111 END+2
*
+TEXTCOLS CUR+2,END,35,9,2    /    asetellaan matriisit vieretysten
*
*1A2B:                                1A2B3C:
*           A   B   C   D   E   F             A   B   C   D   E   F
*  1        7  12   3   3   3   3     1        7  12  33   3   3   3
*  2       10   8   2   2   2   2     2       10   8  22   2   2   2
*  3        5   4   1   1   1   1     3       45  36  13   9   9   9
*  4        5   4   1   1   1   1     4        5   4  11   1   1   1
*  5        5   4   1   1   1   1     5        5   4  11   1   1   1
*  6        5   4   1   1   1   1     6        5   4  11   1   1   1
*
*
*1A2B3C4D:                            1A2B3C4D5E:
*           A   B   C   D   E   F             A   B   C   D   E   F
*  1        7  12  33  51   3   3     1        7  12  33  51  69   3
*  2       10   8  22  34   2   2     2       10   8  22  34  46   2
*  3       45  36  13 153   9   9     3       45  36  13 153 207   9
*  4       80  64 176  19  16  16     4       80  64 176  19 368  16
*  5        5   4  11  17   1   1     5      105  84 231 357  25  21
*  6        5   4  11  17   1   1     6        5   4  11  17  23   1
```

4.3 Valmiiksi annetut luvut

Tähänastisissa esimerkeissä on keskitytty pelkästään *avoimiin ristikoihin*, siis sellaisiin, joissa on annettu vain reunasummat. Ratkaisun helpottamiseksi – toisinaan yksikäsitteistämiseksi – voidaan myös yksi tai useampia ristikkoon sijoitettavista luvuista antaa valmiina (Mustonen 2006a, 1).

Esimerkkinä tällaisen ristikon ratkaisemisesta matriisien avulla tarkastellaan Tehtävää 12 (Mustonen 2006a, 25), johon viitattiin jo aiemmin sivulla 11. Kyseessä on 6×6 -ristikko, jonka 36 luvusta jopa 16 on annettu valmiina. Siitä huolimatta se on vaikeusasteensa (17000) perusteella yksi haastavimpia kuluneen viiden vuoden aikana julkaistuista Survo-ristikoista.

```
*Tehtävä 12 (17000 eli "petomainen")
*
*    A   B   C   D   E   F
*1  10   *  29   *   *   5   83    MIN=1 MAX=36 DISTINCT=1
*2   *  27   *   *   8   *   94
*3  33   *  19  32   *  22  139    OFF-täsmennyksellä luetellaan valmiiksi annetut luvut:
*4   *   *  21  12   *   *  101
*5   *  20   *   *  28   *   86    OFF=10,29,5,27,8,33,19,32,22,21,12,20,28,30,11,35
*6  30   *  11   *   *  35  163
*   93 141  87  99 112 134  666
*
*/ACTIVATE +              /   Alku sujuu hyvin samaan tapaan kuin aiemmin:
*
*P1: 83-10-29-5=39 (3)    /   Selvitetään tarvittavat luvut editoriaalisella laskennalla
+COMB P1 TO P1.TXT        /   P1=PARTITIONS,39,3
+FILE MAKE X1,36,0,X,1    /   Luodaan tyhjä datatiedosto muuttujinaan X1,X2,...,X36
+FILE SAVE P1.TXT TO X1   /   Viedään ositukset tekstitiedostosta datatiedostoon
+XALL X1,X1,36            /   Siirretään tiedot oikeille paikoilleen
+TRANSFORM X1 BY NARY     /   Muunnetaan data BInäärimuotoon: NARY=if(X=MISSING)then(0)else(1)
+MAT SAVE DATA X1 TO P1   /   Talletetaan datatiedosto binäärimatriisiksi
```

Jos kyseessä olisi avoin ristikko, toisin sanoen valmiiksi annettuja lukuja ei olisi, niin pelkästään rivin 1 summalla 83 olisi 16782 erilaista kuuden luvun ositusta. Nyt rivin 1 luvuista kolme on annettu valmiina: 10, 29 ja 5. Kun ne vähennetään, jää selvitettäväksi vain summa 39, jolla on 108 kolmen luvun ositusta. Kun vielä huomioidaan muut valmiiksi annetut luvut, niin mahdollisia osituksia jää 22.

Valmiiksi annetut luvut kuvautuvat COMB-komennon OFF-täsmennyksen ansiosta binäärimatriiseihin $\boldsymbol{P}_1, \boldsymbol{P}_A$ jne. nollina. Nyt uutena asiana mukaan tulee jokaista binäärimatriisia kohden seuraava vaihe, jossa valmiiksi annetut luvut koodataan binäärivektorina:

```
+SAVEP CUR+1,CUR+1,O1.TXT  /  Talletetaan rivin 1 valmiiksi annetut luvut
*10 29 5
+FILE REDUCE X1,*,0        /  Tyhjennetään edellä luodun datatiedoston sisältö
+FILE SAVE O1.TXT TO X1    /  Viedään tiedot tekstitiedostosta datatiedostoon
+XALL X1,X1,36             /  Siirretään tiedot oikeille paikoilleen
+TRANSFORM X1 BY NARY      /  Muunnetaan data BInäärimuotoon
+MAT SAVE DATA X1 TO O1    /  Talletetaan datatiedosto binäärivektoriksi
```

Matriisin $\boldsymbol{P}_1$ ja vektorin $\boldsymbol{O}_1$ yhdistämiseksi ne tehdään yhteenlaskukelpoisiksi samaan tapaan kuin kohdassa 2.5.2 (sivulla 7). Binäärivektoria siis monistetaan matriisin $\boldsymbol{P}_1$ rivien lukumäärän verran. Sen jälkeen ne lasketaan yhteen, jolloin $\boldsymbol{P}_1$ sisältää tiedot sekä COMB:illa löydetyistä osituksista että valmiiksi annetuista luvuista:

```
+MAT DIM P1 /* rowP1=22 colP1=36
+MAT Ones=CON(rowP1,1)
+MAT O1=KRONECKER(Ones,O1)   / *O1~KRONECKER(CON,O1) 22*36
+MAT P1=P1+O1       / *P1~P1+KRONECKER(CON,O1) 22*36
```

Kaikki nämä vaiheet sekä rivien yhdistämisessä tarvittavat ykkösmatriisit tehdään riveille 1–5 ja A–E. Sen jälkeen ne yhdistetään pareittain tuttuun tapaan:

```
+MAT H=VEC(IDN(36,36))'  /  36 x 36 -yksikkömatriisista 1 x 1296 -valintavektori
+MAT P1A=SUB(KRONECKER(P1,OnesA)+KRONECKER(Ones1,2*PA),*,H)
+MAT P2B=SUB(KRONECKER(P2,OnesB)+KRONECKER(Ones2,2*PB),*,H)
+MAT P3C=SUB(KRONECKER(P3,OnesC)+KRONECKER(Ones3,2*PC),*,H)
+MAT P4D=SUB(KRONECKER(P4,OnesD)+KRONECKER(Ones4,2*PD),*,H)
+MAT P5E=SUB(KRONECKER(P5,OnesE)+KRONECKER(Ones5,2*PE),*,H)
```

Yhdistelmien määrät vaihtelevat huomattavasti:

```
+MAT DIM P1A /* rowP1A=176 colP1A=36
+MAT DIM P2B /* rowP2B=3267 colP2B=36
+MAT DIM P3C /* rowP3C=10 colP3C=36
+MAT DIM P4D /* rowP4D=7896 colP4D=36
+MAT DIM P5E /* rowP5E=3483 colP5E=36
*
*Karsimatta: 176*3267*10*7896*3483=1.5813296585856e+014
```

mutta alkukarsinnan jälkeen määrät putoavat jo olennaisesti:

```
+MAT DIM Q1A /* rowQ1A=89 colQ1A=36
+MAT DIM Q2B /* rowQ2B=1342 colQ2B=36
+MAT DIM Q3C /* rowQ3C=10 colQ3C=36
+MAT DIM Q4D /* rowQ4D=3042 colQ4D=36
+MAT DIM Q5E /* rowQ5E=1543 colQ5E=36
*
*Karsimalla: 89*1342*10*3042*1543=5606188010280
*
*5606188010280(10:sanoin)=viisi biljoonaa kuusi sataa kuusi miljardia sata kahdeksan kymmentä ...
*TRIM 67
*viisi biljoonaa kuusi sataa kuusi miljardia sata kahdeksan kymmentä
*kahdeksan miljoonaa kymmenen tuhatta kaksi sataa kahdeksan kymmentä
```

Uudelleenkoodaukset tapahtuvat samalla tavalla kuin aikaisemmin. Sen jälkeen seuraa yhdistämisen ja karsinnan vuorottelua neljään kertaan. Ohessa on poimintoja, joista nähdään, kuinka dimensiot vaihtelevat rajusti eri vaiheissa. Karsinnassa tarvittavien SELECT-ehtojen muodostamisessa on hyötyä 6×6 -koodausmatriisin eri vaiheista, jotka muodostettiin ja näytettiin kohdassa 4.2 (sivulla 19).

```
+MAT Q1A2B=#RAO_KHATRI(Q1A,Q2B)
+MAT DIM Q1A2B /* rowQ1A2B=119438 colQ1A2B=36
[...]
+MAT SAVE DATA Q1A2B TO Q1A2B / SELECT=A*B*C*D*E*F*G*H*I
*  A=N1,16 B=N2,4 C=N3,4 D=N4,4 E=N5,4 F=N7,1 G=N8,1 H=N10,1 I=N12,1
+MAT DIM Q1A2B /* rowQ1A2B=10890 colQ1A2B=36
[...]
+MAT Q1A2B3C=#RAO_KHATRI(Q1A2B,Q3C)
+MAT DIM Q1A2B3C /* rowQ1A2B3C=108900 colQ1A2B3C=36
[...]
+MAT SAVE DATA Q1A2B3C TO Q1A2B3C / SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P
*  A=N1,9  B=N2,3 C=N3,3  D=N4,3  E=N5,3  F=N7,1  G=N8,1  H=N10,1
*  I=N12,1 J=N9,3 K=N11,3 L=N13,1 M=N22,1 N=N33,1 O=N36,1 P=N45,1
+MAT DIM Q1A2B3C /* rowQ1A2B3C=1183 colQ1A2B3C=36
[...]
+MAT Q1A2B3C4D=#RAO_KHATRI(Q1A2B3C,Q4D)
+MAT DIM Q1A2B3C4D /* rowQ1A2B3C4D=3598686 colQ1A2B3C4D=36
```

Tämä on kriittisin kohta. Matriisissa $\boldsymbol{Q}_{1A2B3C4D}$ on noin 3.6 miljoonaa riviä, ja se vie **yli gigatavun** levytilaa. Karsinta on vastaavasti myös kovaa: vaihtoehtoja jää jäljelle alle 300:

```
+MAT SAVE DATA Q1A2B3C4D TO Q1A2B3C4D
*  SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P*Q*R*S*T*U*V*W*X*Y
*  A=N1,4  B=N2,2  C=N3,2  D=N4,2  E=N5,2  F=N7,1  G=N8,1  H=N10,1
*  I=N12,1 J=N9,2  K=N11,2 L=N13,1 M=N22,1 N=N33,1 O=N36,1 P=N45,1
*  Q=N16,2 R=N17,2 S=N19,1 T=N34,1 U=N51,1 V=N64,1 W=N80,1 X=N153,1 Y=N176,1
+MAT DIM Q1A2B3C4D /* rowQ1A2B3C4D=283 colQ1A2B3C4D=36
```

Viimeinen yhdistelyn ja karsinnan vaihe näytetään selvyyden vuoksi kokonaan:

```
+MAT           Q1A2B3C4D5E=#RAO_KHATRI(Q1A2B3C4D,Q5E)
+MAT  DIM      Q1A2B3C4D5E /* rowQ1A2B3C4D5E=436669 colQ1A2B3C4D5E=36
+FILE DEL      Q1A2B3C4D5E
+FILE SAVE MAT Q1A2B3C4D5E TO Q1A2B3C4D5E / TYPE=2 (minimikoko ei enää riitä, luvut isompia)
+FILE MASK     Q1A2B3C4D5E,CASE,1,-
+FILE EXPAND   Q1A2B3C4D5E
+FILE EXPAND   Q1A2B3C4D5E / lisätään tilaa toistamiseen; muuttujia tarvitaan paljon!
*
+VARSTAT Q1A2B3C4D5E / VARSTAT=N1:1,N2:1,N3:1,N4:1,N5:1,N7:1,N8:1,N10:1,N12:1,&
*                             N9:1,N11:1,N13:1,N22:1,N33:1,N36:1,N45:1,&
*                             N16:1,N17:1,N19:1,N34:1,N51:1,N64:1,N80:1,N153:1,N176:1
+VARSTAT Q1A2B3C4D5E / VARSTAT=N21:1,N23:1,N25:1,N46:1,N69:1,N84:1,N105:1,&
*                             N207:1,N231:1,N357:1,N368:1
*   N1=#VAL,1    N2=#VAL,2     N3=#VAL,3     N4=#VAL,4      N5=#VAL,5     N7=#VAL,7
*   N8=#VAL,8   N10=#VAL,10   N12=#VAL,12    N9=#VAL,9     N11=#VAL,11   N13=#VAL,13
*  N22=#VAL,22  N33=#VAL,33   N36=#VAL,36   N45=#VAL,45    N16=#VAL,16   N17=#VAL,17
*  N19=#VAL,19  N34=#VAL,34   N51=#VAL,51   N64=#VAL,64    N80=#VAL,80  N153=#VAL,153
*  N21=#VAL,21  N23=#VAL,23   N25=#VAL,25   N46=#VAL,46    N69=#VAL,69  N176=#VAL,176
*  N84=#VAL,84 N105=#VAL,105 N207=#VAL,207 N231=#VAL,231  N357=#VAL,357 N368=#VAL,368
*.............................
+FILE MASK Q1A2B3C4D5E,CASE,1,A
+FILE MASK Q1A2B3C4D5E,N1,1,-...
+MAT SAVE DATA Q1A2B3C4D5E TO Q1A2B3C4D5E
*  SELECT=A*B*C*D*E*F*G*H*I*J*K*L*M*N*O*P*Q*R*S*T*U*V*W*X*Y*a*b*c*d*e*f*g*h*i*j*k
*  A=N1,1   B=N2,1   C=N3,1   D=N4,1  E=N5,1  F=N7,1  G=N8,1   H=N10,1
*  I=N12,1  J=N9,1   K=N11,1  L=N13,1 M=N22,1 N=N33,1 O=N36,1  P=N45,1
*  Q=N16,1  R=N17,1  S=N19,1  T=N34,1 U=N51,1 V=N64,1 W=N80,1  X=N153,1  Y=N176,1
*  a=N21,1  b=N23,1  c=N25,1  d=N46,1 e=N69,1 f=N84,1 g=N105,1 h=N207,1
*  i=N231,1 j=N357,1 k=N368,1
+MAT DIM Q1A2B3C4D5E /* rowQ1A2B3C4D5E=1 colQ1A2B3C4D5E=36
```

Viimeisen vaiheen 36 SELECT-ehtoa voisi korvata yksinkertaisemmalla tavalla, jossa muodostettaisiin N-muuttujien summa ja vaadittaisiin, että se on 36. Lopputulos on sama: 43**666**9 vaihtoehdon joukosta vain yksi täyttää ehdon.

4.4 Uusia matriisioperaatioita

Vaikeampien ristikoiden ratkaisemisessa käsiteltäviä rivejä voi olla satoja tuhansia, jopa miljoonia. Tällöin Kroneckerin tuloa ja matriisitulkin SUB-funktiota hyödyntävä keino tuhlaa valtavasti tilaa, kun se väliaikaisesti – ennen turhien sarakkeiden karsimista – muodostaa koko Kroneckerin tulon. Kun dimensiot kasvavat tarpeeksi suuriksi, ei tämä keino enää tuota tulosta.

Ehdotuksestani *Seppo Mustonen* ohjelmoi tätä varten Survon matriisitulkkiin operaation `MAT #RAO_KHATRI`, joka muodostaa tarvittavat Kroneckerin tulot operoimalla vain rivisuunnassa. Uutta operaatiota on hyödynnetty runsaasti lukujen 3 ja 4 työkaavioissa. Ilman sitä läheskään kaikkia tilanteita ei olisi saanut ratkaistua.

Samalla syntyi toinenkin matriisioperaatio, `MAT #HADAMARD`, jonka avulla työkaavioista saadaan hieman kompaktimpia. Sinänsä Hadamardin tulot pystyy laskemaan `MAT TRANSFORM` -komennoillakin.

Lähteet

Mustonen, Seppo (2006a). Survo-ristikoista.
`http://www.survo.fi/papers/ristikot.pdf`

Mustonen, Seppo (2006b). Tehtävät 2.10.2006.
`http://www.survo.fi/ristikot/#021006`

Mustonen, Seppo (2007). Uusi Survo-ristikoiden ratkaisuohjelma.
`http://www.survo.fi/arkisto/001175.html`

Mustonen, Seppo (2010). Tehtävät 8.2.2010.
`http://www.survo.fi/ristikot/#080210`