Kimmo Vehkalahti (18.4.2007):

Survo-ristikoita olen harrastanut alusta asti, ts. reilun vuoden. Survoa olen käyttänyt aktiivisesti yli 16 vuoden ajan. Molempiin olen tainnut addiktoitua aika vahvasti, mikä ilmennee seuraavasta.

Yllättäen onnistuin joitakin päiviä sitten kehittämään uuden, mielestäni aika ovelan tavan ratkaista Survo-ristikko. Esittelen sitä seuraavassa melko yksityiskohtaisesti. Kehittämäni tapa vastaa paljolti periaatteita, joita olen soveltanut jo aiemmin etenkin vaikeimmissa (vaikeusaste 2000-17000) ristikoissa, mutta se helpottaa eräitä niissä vaadittavia työvaiheita merkittävästi.

Tarkastellaan esimerkkinä seuraavan ristikon ratkaisemista:

Survo-ristikko 53/2007 (1100 ilman vihjettä)

```
    A  B  C  D
1   *  *  *  *  27
2   *  *  *  *  13        Jätän tässä hyödyntämättä tehtävässä
3   *  *  *  *  53        annetun vihjeen, jolloin kyseessä on
4   *  *  *  *  43        melko vaikea, avoin 4x4-ristikko.
   47 39 29 21
```

Rivien (1,2,3,4) ja sarakkeiden (A,B,C,D) summien partitioita tutkimalla selviää, että joihinkin näistä käy vain muutama vaihtoehto, kun toisiin löytyy monia kymmeniä. Usein tästä voi päätellä, mistä kannattaa aloittaa, jottei ratkaisu käy liian työlääksi. Nyt en kuitenkaan tarvitse tätä tietoa lainkaan.

Ideana on selvittää järjestelmällisesti rivien 1, 2 ja 3 sekä sarakkeiden A, B ja C yhtaikaiset vaihtoehdot tarkastelemalla niitä kokonaisuutena, jolloin ainoa jäljelle jäävä luku (D4) ja samalla koko ristikon ratkaisu määräytyy suoraan. Epäkelvot vaihtoehdot rajautuvat pois tarkastelujen edetessä. Totuttuun tapaan annetun ristikon oletetaan ratkeavan yksikäsitteisesti.

.....

Vähän tarkemmin kuvattuna tapahtuu seuraavaa:

Ensin selvitetään rivi- ja sarakevaihtoehdot 1 ja 2 sekä A ja B, sitten yhdistetyt vaihtoehdot 1A ja 2B. Kaavioissa pisteet symboloivat eri vaihtoehtoja ja o-kirjaimet niiden yhteisiä lukuja A1 ja B2:

```
  1A:                    2B:
     A  B  C  D             A  B  C  D
  1  o  .  .  .  27    1  *  .  *  *  27
  2  .  *  *  *  13    2  .  o  .  .  13
  3  .  *  *  *  53    3  *  .  *  *  53
  4  .  *  *  *  43    4  *  .  *  *  43
    47 39 29 21            47 39 29 21
```

Seuraavaksi tutkitaan 1A:n ja 2B:n yhteisvaihtoehdot, joihin A1:n ja B2:n ohella tulee kaksi muuta yhteistä lukua: A2 ja B1. Näitä merkitään symbolilla + kaaviossa 1A2B:

```
  1A2B:                  3C:                    1A2B3C:
     A  B  C  D             A  B  C  D             A  B  C  D
  1  o  +  .  .  27    1  *  *  .  *  27    1  o  +  +  .  27
  2  +  o  .  .  13    2  *  *  .  *  13    2  +  o  +  .  13
  3  .  .  *  *  53    3  .  .  o  .  53    3  +  +  o  .  53
  4  .  .  *  *  43    4  *  *  .  *  43    4  .  .  .  *  43
    47 39 29 21            47 39 29 21            47 39 29 21
```

Kokonaisuuteen 1A2B yhdistetään vastaavasti muodostettu 3C. Silloin tullaan jo yhdistelmään 1A2B3C, jossa yhteisiä lukuja on 6+3=9 kpl ja joka kattaa miltei koko ristikon. Ratkaisu seuraa välittömästi.

.....

Periaate on siis varsin yksinkertainen: mahdolliset vaihtoehdot "ajetaan nurkkaan", jossa taulukon vapausasteet loppuvat ja jäljelle jää vain yksi, joka osoittautuu ristikon oikeaksi ratkaisuksi.

Käytännön toteutus Survolla nojaa kahteen oivallukseen. Ne ovat:

1) Vaihtoehtojen tarkastelu binäärisinä matriiseina
2) Vaihtoehtojen yhdistely Kroneckerin tulon avulla

Avaan molempia kohtia hieman, jotta ratkaisutapani olisi paremmin ymmärrettävissä. Huomautan jo nyt, että useitakin toteutukseni yksityiskohtia voisi edelleen lyhentää ja optimoida; kuvaan nyt kuitenkin juuri ne keinot joilla sain ratkaisun aikaan.

.....

1) Vaihtoehtojen tarkastelu binäärisinä matriiseina

Kaikki alkoi oikeastaan ristikosta 47/2007 (2260), jonka olin tulostanut paperille ja ottanut mukaan työmatkalleni Puolaan lentokentillä odottelua ym. mahdollisia ajantappoja varten. Kovinkaan pitkälle en päässyt sen kanssa kynällä ja paperilla. En ole tänä vuonna ehtinyt Survo-ristikoita juuri ratkoa, ja aloin jo epäillä taitoni ehtyneen... :)

Palattuani päätin tutkailla eräitä ongelmakohtia Survon avulla. Mietin olisiko jotain matriisioperaatiota tms. joka tuottaisi kahdesta COMB:in tuloslistasta sellaisen, jossa on mahdolliset yhtaikaiset vaihtoehdot. En heti keksinyt, ja suunnittelin jo sukron tekoa. [Aiemmin olen tehnytkin sukron, joka vaiheittain ratkaisee avoimen 4x4-ristikon vaikeusasteeltaan jopa yli 2000, mutta se ainoastaan toistaa ne askeleet, jotka vaaditaan juuri

kyseisen ristikon (Sepon pdf:n 2.6.2006 Tehtävä 11, sivu 24) ratkaisussa, ja jotka olin siis ensin kulkenut - sekä Survolla että kynällä ja paperilla. Tuo sukro ei siis auta tässä yhtään.]

Jotenkin vain sain päähäni ajatella yhtaikaisten vaihtoehtojen tarkastelua binäärimuodossa, ja melkein samantien keksin mitä ainesosia keitokseen tarvitaan. Peruslistat tuottaa tietenkin COMB, esimerkiksi tässä rivin 2 osalta:

```
COMB P2 CUR+2 / P2=PARTITIONS,13,4 DISTINCT=1

Partitions 4 of 13: N[P2]=3
1 2 3 7
1 2 4 6
1 3 4 5
```

Listan binäärimuodolla tulen tarkoittamaan seuraavaa:

```
MATRIX P2
///       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
  1       1  1  1  0  0  0  1  0  0  0  0  0  0  0  0  0
  2       1  1  0  1  0  1  0  0  0  0  0  0  0  0  0  0
  3       1  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0
```

Otsikointi paljastaa, että jälkimmäisen muodon tuottaa Survon matriisitulkki, mutta välissä tarvitaan paria muuta toimintoa. Yksi tapa rakentaa kyseinen matriisi Survossa on:

```
COMB P2 TO P53_2.TXT / P2=PARTITIONS,13,4 DISTINCT=1     /  (1)
FILE MAKE X53_2,16,0,X,1                                 /  (2a)
FILE SAVE P53_2.TXT TO X53_2                             /  (2b)
XALL X53_2,X1,16                                         /  (3)
TRANSFORM X53_2 BY if(X=MISSING)then(0)else(1)           /  (4)
MAT SAVE DATA X53_2 TO P2                                /  (5a)
MAT CLABELS NUM(1) TO P2                                 /  (5b)
MAT LOAD P2 11 CUR+2 / Selitykset kohtiin (1)-(5) alla.  /  (5c)
```

(1): talletetaan em. lista tekstitiedostoksi ilman otsikoita.

(2a,b): luodaan 16 N1-muuttujan data ja siirretään tiedot sinne:

```
  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12 X13 X14 X15 X16
   1   2   3   7   -   -   -   -   -   -   -   -   -   -   -   -
   1   2   4   6   -   -   -   -   -   -   -   -   -   -   -   -
   1   3   4   5   -   -   -   -   -   -   -   -   -   -   -   -
```

Tässä muodossa tämä ei vielä hyödytä yhtään mitään, mutta sitten tuleekin yhtäkkiä käyttöä operaatiolle, joka on odottanut kohta 10 vuotta, että pääsisi taas tositoimiin! Tein 9.5.1997 modulin XALL (etymologia on: "X ALL that apply", rastita kaikki kyseeseen tulevat kohdat [kyselylomakkeen osiossa]) Nokian USA:ssa tekemän kyselytutkimuksen aineistojen muokkauksen yhteydessä. Lomakkeessa käytetty koodaustapa oli juuri tämä: muuttujat vastasivat mm. eri harrastuksia, joita sai rastittaa niin monta kuin halusi.

Tilan säästämiseksi tutkimusyritys oli koodannut tiedot samalla tavalla kuin COMB ne tuottaa. Tutkimusavustajamme oli ihmeissään, kunnes autoin häntä ohjelmoimalla tuon merkillisen XALL-modulin. Se ei ole kuulunut Survon vakiokalustoon, mutta on ollut jakelussa aikoinaan Survo-Käyttäjäyhdistyksen SURVOTUT-levykkeillä 4 ja 5.

(3): siirretään luvut oikeille paikoilleen:

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
1	2	3	-	-	-	7	-	-	-	-	-	-	-	-	-
1	2	-	4	-	6	-	-	-	-	-	-	-	-	-	-
1	-	3	4	5	-	-	-	-	-	-	-	-	-	-	-

Minulla XALL on ollut käytössä myös SURVO MM -aikakaudella, sillä olen demonstroinut välillä tuota erikoista muunnosta tilastollisen tietojenkäsittelyn perusteet -kurssillani (2002-2007). Mihinkään sen hyödyllisempään en ole XALL:ia 2000-luvulla käyttänytkään.

(4): muunnetaan koko data binääriseen muotoon:

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

(5a-c): talletetaan data matriisiksi ja vaihdetaan sarakeotsikot:

MATRIX P2

///	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
2	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
3	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Sarakeotsikot viittaavat siis ristikkoon tuleviin lukuihin, ja riviotsikot numeroivat ristikon riville 2 sopivat 3 vaihtoehtoa. Aivan vastaavasti kuin COMBin alkuperäisestä tuloslistasta, nähdään tästäkin, että vain pienet luvut käyvät (summahan on 13), ja että niistä yksi tulee joka tapauksessa olemaan ykkönen. Isoin ero COMBin listaan on sarakedimensio, joka on (4x4-ristikoilla) 16 riippumatta siitä, montaako lukua riville tai sarakkeelle haetaan. Tämä tekee binäärimatriisien jatkokäytön COMB-listoja yksinkertaisemmaksi.

.....

2) Vaihtoehtojen yhdistely Kroneckerin tulon avulla

Binääritaulukko kirvoitti heti kysymyksen, saisiko sitä käyttämällä vertailtua eri vaihtoehtoja toisiinsa. Koetin löytää tehtävään jotain sopivaa matriisioperaatiota. Lopulta keksin, että siinä voi hyödyntää ns. Kroneckerin tuloa. Näytän pienen esimerkin avulla, mistä on kyse ylipäätään (esimerkin matriisit eivät liity Survo-ristikoihin).

Tarkastellaan kahta binääristä matriisia A (3x4) ja B (2x4):

MATRIX A

///	1	2	3	4
A1	1	1	0	0
A2	1	0	1	0
A3	1	0	0	1

MATRIX B

///	1	2	3	4
B1	0	1	1	0
B2	1	1	0	0

(Binäärisyys ei ole mikään edellytys vaan luvut voivat olla mitä tahansa. Havainnollisuuden vuoksi pitäydyn nollissa ja ykkösissä.)

Rivien ja sarakkeiden sisätuloilla operoiva tavanomainen matriisitulo

onnistuu, jos B transponoidaan, koska silloin A:n ja B':n dimensiot ovat yhteensopivat (3x4 ja 4x2). Näin saataisiin matriisi C:

```
MAT C=A*B'          / *C~A*B' 3*2
MAT LOAD C 11 CUR+1
MATRIX C             C.stä nähdään sinänsä kiinnostavia asioita
A*B'                 A:n ja B:n relaatioista: riveillä A1 ja B2 on
///      B1 B2       molemmilla 2 ykköstä samoissa kohdissa, kun
A1        1  2       taas riveillä A3 ja B1 on joka kohdassa eri
A2        1  1       luku. Muissa yhdistelmissä esiintyy ykköspari
A3        0  1       jossakin (tasan yhdessä) kohdassa.
```

Näistä huolimatta tästä ei kuitenkaan päästä juurikaan eteenpäin: 3*4+2*4=20 luvusta on saatu 3*2=6 luvun yhteenveto (kuten toki voi tulosummamatriisilta odottaakin), mutta se mitä varsinaisesti tarvitaan on enemmänkin "yksilökohtaista" yhteenvetoa matriisien A ja B alkioista. Kerrassaan mainion välineen tähän tarkoitukseen tarjoaa Kroneckerin tuloksi kutsuttu matriisioperaatio.

Kroneckerin tulo onnistuu aina, koska siinä matriisien dimensiot saavat olla mitä tahansa. Kyseessä on alkiokohtainen tulo, jossa jokaisella A:n alkiolla kerrotaan jokainen B:n alkio. Tulosmatriisi kasvaa äkkiä: tässäkin pikkuesimerkissä sen kooksi tulee jo 6x16:

```
MAT D=KRONECKER(A,B)          / *D~KRONECKER(A,B) 6*16
MAT LOAD D 111 CUR+1
MATRIX D
KRONECKER(A,B)
///      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
A1*B1      0   1   1   0   0   1   1   0   0   0   0   0   0   0   0   0
A1*B2      1   1   0   0   1   1   0   0   0   0   0   0   0   0   0   0
A2*B1      0   1   1   0   0   0   0   0   0   1   1   0   0   0   0   0
A2*B2      1   1   0   0   0   0   0   0   1   1   0   0   0   0   0   0
A3*B1      0   1   1   0   0   0   0   0   0   0   0   0   0   1   1   0
A3*B2      1   1   0   0   0   0   0   0   0   0   0   0   1   1   0   0
```

Kroneckerin tulo on Survon matriisitulkin "otsikkoaritmetiikan" juhlaa: rivi- ja sarakeotsikot dokumentoivat matriisin syntytavan täydellisesti!

Kun 20 lukua äsken tiivistyi kuuteen, saadaan nyt (3*2)*(4*4)=96 lukua! Matriisin D voi hahmottaa esim. niin, että A:n jokaisen luvun tilalle on pantu ko. luvulla kerrottu B (12*8=96). [Kroneckerin tulon erikoistapaus on selvästikin matriisin kertominen pelkällä skalaarilla.]

Nyt riittää keskittyä vastaavuuksiin samannumeroisten sarakkeiden kesken. Suurin osa matriisista D on siten tässä yhteydessä turhaa. Kiinnostavia ovat vain sarakkeet 1*1, 2*2, 3*3 ja 4*4. Ne saadaan siististi erilleen valintavektorilla E, jossa on ykköset näillä kohdin ja muuten nollaa:

```
MAT E=VEC(IDN(4,4))'  /  yksikkömatriisi muutettuna rivivektoriksi
MAT LOAD E 111 CUR+1
MATRIX E
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
```

Kiinnostava osa D:stä irtoaa sen jälkeen vaivatta:

```
MAT F=SUB(D,*,E)      /  kaikki rivit, mutta vain valitut sarakkeet
MAT LOAD F 111 CUR+1
MATRIX F
SUB(KRONECKER(A,B),*,E)
///      1*1 2*2 3*3 4*4    Rivi- ja sarakeotsikoiden automaattisen,
A1*B1      0   1   0   0    operaatioita myötäilevän periytymisen
A1*B2      1   1   0   0    ansiosta tässäkään ei tarvitse tuhlata
A2*B1      0   0   1   0    aikaa pohtimalla, tulivatko nyt varmasti
A2*B2      1   0   0   0    oikeat sarakkeet kyytiin! (Eräillä muilla
A3*B1      0   0   0   0    matriiseihin erikoistuneilla ohjelmilla
A3*B2      1   0   0   0    tällainen on silkkaa hapuilua...)
```

F:stä nähdään samat asiat kuin edellä mutta tarkemmin: lähtökohtana olleiden matriisien A ja B riveillä A1 ja B2 on molemmilla 2 ykköstä samoissa kohdissa, ja nuo kohdat ovat 1 ja 2. Riveillä A3 ja B1 ei ole yhtään lukua samoissa kohdissa, minkä C kertoi ihan yhtä hyvin. Muissa yhdistelmissä esiintyy ykköspari tasan yhdessä kohdassa, ja nuo kohdat paljastuvat tuosta yksikäsitteisesti.

Yleinen viisaus Survo-ristikoita ajatellen onkin, että kannattaa keskittyä pääasiassa yhteisiin lukuihin, sillä niiden avulla saa yksikäsitteisempiä lopputuloksia kuin tarkastelemalla toisensa poissulkevia rivejä tai sarakkeita (mikä on sisänsä myös aivan mahdollista ja joissain erikoistapauksissa jopa suositeltavaa).

.....

Nyt ryhdyn esittelemään ratkaisutapaani tarkemmin.

Aluksi rivin 1 ja sarakkeen A listoista tehdään edellä kuvatulla tavalla binääriset matriisit P1 (61x16) ja PA (27x16), molemmissa alkiot {0,1}. Sovelletaan niihin Kroneckerin tuloa:

```
MAT DIAG=VEC(IDN(16,16))'              / valintavektori (1x256)
MAT P1A=SUB(KRONECKER(P1,PA),*,DIAG)   / Kroneckerin parhaat palat
MAT DIM P1A /* rowP1A=1647 colP1A=16   / 61*27=1647
```

P1A on tietenkin myös binäärimatriisi. Suurin osa siitä on nollaa:

```
MATRIX P1A
///       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
1*2       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
1*3       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
1*4       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
...
1*23      0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
2*2       0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
2*3       0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
...
5*12      0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1
5*13      0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1
...
61*25     0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
61*26     0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
```

Tästä massiivisesta alkukarsinnasta pääsevät jatkoon vain ne rivit, joissa on tasan 1 ykkönen (yhteisen luvun A1 osoitin), esimerkiksi yllänäkyvistä kaikki muut paitsi "5*12" ja "5*13". Muut karsitaan kierrättämällä matriisi datatiedoston kautta. Rivejä jää 809:

```
FILE DEL P1A               / mahdollinen aiempi data tuhotaan
FILE SAVE MAT P1A TO P1A   / matriisi dataksi, muuttujat: TYPE=1
FILE MASK P1A,CASE,1,-     / riviotsikot piiloon laskelmilta
VARSTAT P1A,SUM:1,SUM      / summa kertoo ykkösten lukumäärän rivillä
FILE MASK P1A,CASE,1,A     / riviotsikot takaisin (niitä tarvitaan!)
FILE MASK P1A,SUM,1,-      / summa piiloon (sitä ei haluta matriisiin)
.............................
MAT SAVE DATA P1A TO P1A   / IND=SUM,1 eli ne joissa tasan 1 ykkönen
MAT CLABELS NUM(1) TO P1A  / sarakeotsikot numeroiksi 1-16
MAT DIM P1A /* rowP1A=809 colP1A=16
MAT PA=2*PA                / uudelleenkoodaus (ks. alla!)
.............................
```

Yksikäsitteiseen loppuratkaisuun pääsemiseksi on pidettävä kirjaa siitä, kumpaan riviin tai sarakkeeseen mikäkin yhdistelmän luku kuuluu. Tätä varten PA koodataan nyt uudelleen kertomalla sen kaikki alkiot kahdella, jolloin ne ovatkin {0,1}:n sijasta {0,2}.

Tämän jälkeen suoritetaan varsinainen matriisien yhteenlaskuoperaatio: P1A:n rivit pannaan yksitellen uusiksi laskemalla P1:n ja PA:n rivit yhteen P1A:n (Kroneckerin tulosta saatujen) riviotsikoiden mukaisesti. Käytännössä se tarkoittaa 809 MAT-komennon aktivointia:

```
MAT P1A(001,1)=P1(01,*)+PA(02,*) / Näiden komentojen kasaamisen jälkeen
MAT P1A(002,1)=P1(01,*)+PA(03,*) / jatkuva aktivointi (F2-Esc) hoitaa
MAT P1A(003,1)=P1(01,*)+PA(04,*) / homman muutamassa sekunnissa.
...
MAT P1A(809,1)=P1(61,*)+PA(26,*)
```

Kyseisen hässäkän rakentaa helposti blokkisiirroilla ja komennoilla SET, COUNT, INSERT, DELETE, REPLACE, FORM jne., mutta jossain vaiheessa kyllästyin manuaaliseen toistoon ja tein pienen sukron avukseni (se olikin tällä kertaa ainoa!). Sivuutan ohjelmakoodin tässä yhteydessä, mutta sukro tekee nopeasti ja nöyrästi tarvittavat vaiheet, kun komennan

```
/PSUM P1A P1 PA
```

ja kirjoittaa puolestaan komentorivin alapuolelle valmiin komennon

```
LOADM P1A 11 CUR+1 / (809 lines) LIMITS=0,1,2,3 SHADOWS=0,4,7,5
```

jolla tulosmatriisia voi ihailla toimituskentässä värikkäässä muodossa. Tyydytään tässä mustavalkoiseen esitykseen, josta nähdään yhtä hyvin, että talteen on saatu enemmän tietoa P1:n ja PA:n vaihtoehdoista:

```
MATRIX P1A
///       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
1*2       1  1  2  0  0  0  0  1  0  0  0  0  2  0  2  3
1*3       1  1  0  2  0  0  0  1  0  0  0  2  0  0  2  3
1*4       1  1  0  2  0  0  0  1  0  0  0  0  2  2  0  3
...
1*23      1  1  0  0  0  0  0  1  2  2  0  2  0  0  0  3
2*2       1  1  2  0  0  0  0  0  1  0  0  0  2  0  3  2
2*3       1  1  0  2  0  0  0  0  1  0  0  2  0  0  3  2
...
61*25     0  0  0  0  1  1  1  0  3  0  2  2  0  0  2  0
61*26     0  0  0  0  1  1  1  0  3  0  2  0  2  2  0  0
```

P1A:n alkiot {0,1,2,3} kuvaavat nyt sarakeindeksejään vastaavien lukujen sijaintia ristikossa:

```
  0 = ko. luku ei esiinny rivillä 1 eikä sarakkeessa A
  1 = ko. luku esiintyy rivillä 1
  2 = ko. luku esiintyy sarakkeessa A
  3 = ko. luku esiintyy rivillä 1 sarakkeessa A (A1)
```

Koodaus on siis yksityiskohtaisempi kuin alussa esitetyt kaaviot.

```
  1A: (a)              1A: (b)                      1A: (c)
     A  B  C  D           A    B    C    D           A  B  C  D
  1  o  .  .  .  27    1  [ 3] [ 1] [ 1] [ 1] 27    1  9  7  6  5  27
  2  .  *  *  *  13    2  [ 2] [ 0] [ 0] [ 0] 13    2 11  *  *  *  13
  3  .  *  *  *  53    3  [ 2] [ 0] [ 0] [ 0] 53    3 13  *  *  *  53
  4  .  *  *  *  43    4  [ 2] [ 0] [ 0] [ 0] 43    4 14  *  *  *  43
    47 39 29 21           47   39   29   21          47 39 29 21
```

Vasemmanpuoleinen kaavio (a) on sama kuin alussa, mutta (b):ssä lukuina ovatkin P1A:n alkiot. Koska ne ovat pikemminkin osoittimia kuin varsinaisia lukuja, olen merkinnyt ne hakasulkuihin.

Kaavio (c) esittää mahdollisen (osa)realisaation luvuilla, jotka on poimittu P1A:n viimeisen rivin ("61*26") sarakeindekseistä. On huomattava, että vaikka rivin 1 ja sarakkeen A summat pätevät, niin (c) ei kerro mitään lukujen 5-7 ja 11-14 oikeasta järjestyksestä (puhumattakaan siitä miten loput luvut sijoittuisivat ristikkoon).

Luku 9 on ainoa, jonka paikka on tarkasti määrätty, mutta mikään ei takaa, johtaako juuri tämä P1A:n viimeisen rivin valikoima ristikon oikeaan ratkaisuun. Todennäköisesti ei, onhan se vain yksi 809:stä.

.....

Tulokset eivät näytä tässä vaiheessa hääveiltä, mutta minulla oli sellainen aavistus, että kannattaisi jatkaa ja katsoa miten käy...

.....

Aivan vastaavasti yhdistetään P2 ja PB ja saadaan vastaavanlainen yhteisvalikoima P2B, jonka alkiot ovat samalla tavoin {0,1,2,3}.

Ratkaiseva askel on yhdistää P1A ja P2B muodostamalla P1A2B, joka sisältää tiedot ristikon kahden rivin ja kahden sarakkeen luvuista. Sama periaate pätee edelleen, mutta koodausta on vain laajennettava. Toisaalta yhdistämisissä selvitäänkin yllättäen vähemmällä.

Näytän heti aluksi, että tässä tullaan pääsemään jo varsin pitkälle:

```
  1A2B: (a)            1A2B: (b)                    1A2B: (c)
     A  B  C  D           A    B    C    D           A  B  C  D
  1  o  +  .  .  27    1  [ 7] [12] [ 3] [ 3] 27    1 10  8  4  5  27
  2  +  o  .  .  13    2  [10] [ 8] [ 2] [ 2] 13    2  7  3  1  2  13
  3  .  .  *  *  53    3  [ 5] [ 4] [ 1] [ 1] 53    3 14 13  *  *  53
  4  .  .  *  *  43    4  [ 5] [ 4] [ 1] [ 1] 43    4 16 15  *  *  43
    47 39 29 21           47   39   29   21          47 39 29 21
```

Kaavio (a) on jälleen sama kuin alussa, ja (b):ssä lukujen tilalla ovat osoitinluonteiset koodit, joita on nyt kolmen sijasta yhdeksän. Kaavio (c) esittää taas yhden mahdollisen osarealisaation, josta enää puuttuvat luvut 6, 9, 11 ja 12. Kokeilemalla selviää helposti, ettei oikea ratkaisu aukene tästä. Esimerkiksi sarakkeelle C tarvittaisiin yhteensä 29-4-1=24, mutta se ei onnistu em. neljällä luvulla.

Ennen kuin päästään loppunäytökseen, katsotaan tarkemmin kaaviossa (b) esiintyviä koodeja. Niihin päädytään, kun kirjanpito mukautetaan ottamaan huomioon yhtaikaa 1A:n ja 2B:n asetelmat. Koodaustapoja on varmasti monenlaisia, mutta pähkäiltyäni asiaa päädyin tällaiseen:

```
MAT TRANSFORM P1A BY 2*X#+1   /  {1,3,5,7}
MAT TRANSFORM P2B BY 2^X#     /  {1,2,4,8}
```

Molemmissa alkiot ovat alunperin {0,1,2,3}. Jos näitä kerrottaisiin keskenään, saataisiin arvoja {0,1,2,3,4,6,9}. Niistä {4,6,9} ovat loogisesti mahdottomia. Koodaus jähmettyisi siis aikaisempaan, eikä kantaisi informaatiota uudesta, yhdistetystä tilanteesta. Syynä on ennen kaikkea nolla, joka kerrottuna millä tahansa tuottaa nollan.

Kaaviona ongelma näkyy selvästi: (b):n tapaiset kaaviot kerrottuna keskenään (alkioittain) antavat oikeassa reunassa olevan tuloksen:

```
      A    B    C    D      A    B    C    D      A    B    C    D
1  [ 3] [ 1] [ 1] [ 1] * [ 0] [ 2] [ 0] [ 0] = [ 0] [ 2] [ 0] [ 0]
2  [ 2] [ 0] [ 0] [ 0] * [ 1] [ 3] [ 1] [ 1] = [ 2] [ 0] [ 0] [ 0]
3  [ 2] [ 0] [ 0] [ 0] * [ 0] [ 2] [ 0] [ 0] = [ 0] [ 0] [ 0] [ 0]
4  [ 2] [ 0] [ 0] [ 0] * [ 0] [ 2] [ 0] [ 0] = [ 0] [ 0] [ 0] [ 0]
```

Ilman muunnosta saataisiin vain tieto uusista yhteisistä luvuista A2 ja B1, mutta menetettäisiin siihenastinen kirjanpito. Ilmeisesti edellä esitetty summausmenetelmä toisi tiedot takaisin, mutta ne eivät olisi riittävän yksikäsitteisiä. (Sitäpaitsi summaus on kömpelöä, eikä sitä jatkossa enää tarvita. Ehkei sitä tarvittaisi alussakaan, mutten ala nyt muuttaa toimivaa ratkaisuani kesken raportoinnin.)

Pelkkä nollan korjaaminen ei auta. Jos vaikka lisätään molempiin 1, ts. käytetään koodeja {1,2,3,4}, saadaan kyllä selvästi parempaa, mutta ongelmana on edelleen kaksikäsitteisyys. Yhdistelmiä 1A ja 2B ei kyettäisi erottamaan toisistaan:

```
      A    B    C    D      A    B    C    D      A    B    C    D
1  [ 4] [ 2] [ 2] [ 2] * [ 1] [ 3] [ 1] [ 1] = [ 4] [ 6] [ 2] [ 2]
2  [ 3] [ 1] [ 1] [ 1] * [ 2] [ 4] [ 2] [ 2] = [ 6] [ 4] [ 2] [ 2]
3  [ 3] [ 1] [ 1] [ 1] * [ 1] [ 3] [ 1] [ 1] = [ 3] [ 3] [ 1] [ 1]
4  [ 3] [ 1] [ 1] [ 1] * [ 1] [ 3] [ 1] [ 1] = [ 3] [ 3] [ 1] [ 1]
```

Koodit on siis valittava niin, ettei eri yhdistelmien koodeilla ole (triviaalia ykköstä lukuunottamatta) yhteisiä tekijöitä:

```
      A    B    C    D      A    B    C    D      A    B    C    D
1  [ 7] [ 3] [ 3] [ 3] * [ 1] [ 4] [ 1] [ 1] = [ 7] [12] [ 3] [ 3]
2  [ 5] [ 1] [ 1] [ 1] * [ 2] [ 8] [ 2] [ 2] = [10] [ 8] [ 2] [ 2]
3  [ 5] [ 1] [ 1] [ 1] * [ 1] [ 4] [ 1] [ 1] = [ 5] [ 4] [ 1] [ 1]
4  [ 5] [ 1] [ 1] [ 1] * [ 1] [ 4] [ 1] [ 1] = [ 5] [ 4] [ 1] [ 1]
```

Nyt 1A2B:n taustalla olevat 1A ja 2B kyetään erottamaan toisistaan, ts. vaivalla koottua informaatiota ei hukata. Rivien 3 ja 4 sekä sarakkeiden C ja D koodaus ei tässä vaiheessa ole yksikäsitteistä, koska informaatio perustuu vasta kahteen riviin ja sarakkeeseen.

.....

Katsotaan millaisia vaiheita P1A2B:n luonti ja karsinta sisältää. Aluksi P1A:n ja P2B:n riviotsikot resetoidaan, jotta muodostettavan matriisin otsikot pysyvät lukukelpoisina:

```
MAT RLABELS NUM(1) TO P1A
MAT RLABELS NUM(1) TO P2B
```

Nyt riittää laskea Kroneckerin tulo uudelleenkoodatuista matriiseista. Sarakkeiden poimintaan käytetään edellä luotua valintavektoria DIAG:

```
MAT P1A2B=SUB(KRONECKER(P1A,P2B),*,DIAG)    / kestää n. 14 sekuntia...
MAT DIM P1A2B /* rowP1A2B=99507 colP1A2B=16 / ...ja syykin on selvä!
```

Yhdistelmiä on lähes sata tuhatta! Hetkellisesti (ennen kuin turhat sarakkeet karsiutuvat) matriisi P1A2B vie yli 200 megatavua levytilaa, mutta siitähän nykyään harvemmin on enää pulaa.

Koodaustavasta riippumatta osa yhdistelmistä on mahdottomia, esim. "luku esiintyy rivillä 1 ja rivillä 2" tai "luku on A1 ja B2" jne. Näitä vaihtoehtoja edustavat koodauksessani luvut 6, 14, 20, 24, 28, 40 ja 56, eivätkä ne siis esiinny edelläolevassa kaaviossa (b). Rajataan ne pois noiden 99507:n joukosta laskemalla esiintymien lukumääriä. Jännittävää on, kuinka monta vaihtoehtoa jää jäljelle:

```
FILE DEL P1A2B
FILE SAVE MAT P1A2B TO P1A2B   / TYPE=1
FILE MASK P1A2B,CASE,1,-       / passivoidaan CASE ehtojen ajaksi
VARSTAT P1A2B,z1:1,#VAL,6      / yksikin 6 riittää hylkäykseen
VARSTAT P1A2B,z2:1,#VAL,14,56  / samoin näissä (14,20,24,28,40,56)
```

Suurin osa näyttää putoavan "semifinaalikarsinnassa" pois:

```
.....................................................................

TAB P1A2B CUR+3 / VARIABLES=z2,z1 CHI2=- LABELS=0
                  z1=*c z2=*c *c=0,0(OK!),99(ei_käy)

       z2    OK! ei_käy
z1     **
OK!         1108  20252
ei_käy      9506  68641

.....................................................................
```

Tiukennetaan ehtoja vaatimalla, että yhteiset luvut A1, B2, A2 ja B1 ovat yksikäsitteisiä, ks. kaavio (b) edellä:

```
VARSTAT P1A2B,z3:1,#VAL,7  / pitää siis olla tasan 1 seiska
VARSTAT P1A2B,z4:1,#VAL,8  / (vastaavasti nämä)
VARSTAT P1A2B,z5:1,#VAL,10 /
VARSTAT P1A2B,z6:1,#VAL,12 /
```

```
.....................................................................
```

Tarkistetaan tilanne luomalla indikaattorimuuttuja OK, joka kokoaa kaikki edellä asetetut ehdot yhteen:

```
VAR OK=1 TO P1A2B / SELECT=a*b*c*d*e*f eli kaikki ehdot yhtaikaa:
                           a=z1,0 b=z2,0 c=z3,1 d=z4,1 e=z5,1 f=z6,1
................
```

```
STAT P1A2B CUR+2  / VARS=OK IND=OK

Basic statistics: P1A2B N=7
Variable: OK       ~1
Constant= 1

.....................................................................
```

Vaikuttaa tiukalta kilvalta: vain 7 "kilpailijaa" selviytyi finaaliin!
Rekisteröidään näiden tiedot päivittämällä matriisi P1A2B:

```
FILE MASK P1A2B,CASE,1,A     / CASE takaisin aktiiviseksi
FILE MASK P1A2B,z1,1,-...    / indikaattorit piiloon
.............................
MAT SAVE DATA P1A2B TO P1A2B / IND=OK
MAT CLABELS NUM(1) TO P1A2B
MAT DIM P1A2B /* rowP1A2B=7 colP1A2B=16
```

Tässä ovat finalistit:

```
MAT LOAD P1A2B 11 CUR+1
MATRIX P1A2B
///       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
517*116   2  3  2 10  8  3 12  1  1  1  1  7  4  4  5  5
518*103   2  3  2  8 10  3 12  1  1  1  1  7  4  5  4  5
531*106   2  3  2  8 10  3  1 12  1  1  7  1  4  4  5  5
664*62    2  2  3  8  3 10  1 12  1  1  7  4  1  5  4  5
678*66    2  2  3  8  3 10  1  1 12  7  1  4  1  4  5  5
731*18    2  2  8  3  3 12 10  1  1  1  1  7  5  4  5  4
756*20    2  2  8  3  3  1 10 12  1  7  1  1  4  5  4  5
```

Edellä esitetty umpikujaan johtava kaavio (c) perustuu viimeiseen riviin ("756*20"). Voin paljastaa, että tuleva "voittaja" lymyilee taulukon keskivaiheilla... :)

```
.....................................................................
```

Tunnustan auliisti, että ollessani tässä vaiheessa en ollut yhtään varma, johtaako tämä touhu toivottuun tulokseen vai tuleeko karvas finaalitappio. Ei auttanut kuin jatkaa...

```
.....................................................................
```

P3C (238 vaihtoehtoa) tehdään ihan samalla tavalla kuin P1A ja P2B. Sen jälkeen P3C yhdistetään äsken aikaansaatuun 7-riviseen P1A2B:hen.

Koodausta on jälleen laajennettava vastaavasti kuin edellä. Keinoni olivat loppua kesken, sillä en keksinyt kätevää funktiota muunnosta varten. Niinpä turvauduin ykkösen lisäksi (alku)lukuihin 11, 13 ja 17 ja tein matriisiin P3C "ABC-muunnoksen" datatiedoston puolella:

```
FILE DEL TMP
FILE SAVE MAT P3C TO TMP / TYPE=1
..................
TRANSFORM TMP BY if(X=0)then(01)else(A) / VARS=ALL,-CASE
               A=if(X=1)then(11)else(B)
               B=if(X=2)then(13)else(C)
               C=if(X=3)then(17)else(X)
.....................................
MAT SAVE DATA TMP TO P3C
MAT RLABELS NUM(1) TO P3C
.....................................
```

Yhdistäminen sujuu täsmälleen samoin kuin edellä, ja tulos on P1A2B3C (1666 vaihtoehtoa). Tilanne käy yhä jännittävämmäksi: ehtojen tulisi rajata pois kaikki paitsi ristikon oikea ratkaisu!

.....

Nyt pitää olla tarkkana, sillä pienten lukujen takia indikaattorit on perustettava ensin ja passivoitava samantien, jottei niissä olevia lukuja lasketa VARSTATeissa mukaan! (Toki MASK-täsmennyksellä voisi rajata käsiteltävät muuttujat, mutta FILE MASK tuli ensin mieleen.)

```
FILE EXPAND P1A2B3C / tilaa tarvitaan lisää ja sitähän saadaan tällä
VAR z1:1,z2:1,z3:1,z4:1,z5:1,z6:1,z7:1,z8:1,z9:1 TO P1A2B3C
  z1=MISSING z2=z1 z3=z2 z4=z3 z5=z4 z6=z5 z7=z6 z8=z7 z9=z8
FILE MASK P1A2B3C,z1,1,-... / passivoidaan z1-z9
.........................

VARSTAT P1A2B3C,z1,#VAL,22     / mahdottomat yhdistelmät
VARSTAT P1A2B3C,z2,#VAL,33,34
VARSTAT P1A2B3C,z3,#VAL,51,52
VARSTAT P1A2B3C,z4,#VAL,65,204

VARSTAT P1A2B3C,z5,#VAL,39     / joukko yksikäsitteisiä
VARSTAT P1A2B3C,z6,#VAL,44
VARSTAT P1A2B3C,z7,#VAL,55
VARSTAT P1A2B3C,z8,#VAL,26
VARSTAT P1A2B3C,z9,#VAL,1

.........................
VAR OK=1 TO P1A2B3C / SELECT=a*b*c*d*e*f*g*h*i
                      a=z1,0 b=z2,0 c=z3,0 d=z4,0
                      e=z5,1 f=z6,1 g=z7,1 h=z8,1 i=z9,1
.................
FILE MASK P1A2B3C,CASE,1,A
FILE MASK P1A2B3C,OK,1,-
FILE SHOW P1A2B3C               /  tässä vaiheessa siis N=1666...
.........................
MAT SAVE DATA P1A2B3C TO P1A2B3C /  IND=OK       ...nyt selviää...
MAT CLABELS NUM(1) TO P1A2B3C
MAT DIM P1A2B3C /* rowP1A2B3C=1 colP1A2B3C=16
```

Vain yksi rivi jää jäljelle! Mikä se on? Onko sillä lopulta mitään tekemistä Survo-ristikon ratkaisun kanssa?? (En edelleenkään ollut ihan varma, oliko proseduurini pätevä...)

```
MAT LOAD P1A2B3C 11 CUR+1 / tutustutaan voittajakandidaattiin
MATRIX P1A2B3C
///       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
4*46     26  2  3  8 39 10  1 12 11 13  7  4 17  5 44 55
```

(Ensimmäinen ajatus oli: "Ei hyvältä näytä!" Isot luvut vaikuttivat jotenkin oudoilta...)

Eikä kuin sijoittamaan lukuja oikeille paikoilleen! Ja tulos on...

1A2B3C: (a)

	A	B	C	D	
1	o	+	+	.	27
2	+	o	+	.	13
3	+	+	o	.	53
4	.	.	.	*	43
	47	39	29	21	

1A2B3C: (b)

	A	B	C	D	
1	[7]	[12]	[39]	[3]	27
2	[10]	[8]	[26]	[2]	13
3	[55]	[44]	[17]	[11]	53
4	[5]	[4]	[13]	[1]	43
	47	39	29	21	

1A2B3C: (c)

	A	B	C	D	
1	11	8	5	3	27
2	6	4	1	2	13
3	16	15	13	9	53
4	14	12	10	7	43
	47	39	29	21	

Kyllä, finaali on päättynyt onnellisesti: kaaviossa (c) on Survo-ristikon 53/2007 ainoa oikea ratkaisu. Ratkaisutapani siis toimii!

(Tarkistin vielä neuroottisesti jokaisen rivin ja sarakkeen yksittäin kosketuslaskennalla, ennen kuin lähetin Sepolle tiedonannon...)

* * *

Ilmeisesti ratkaisukeinoni on melko pätevä, ja jopa yleiskäyttöinen, ainakin periaatteessa... Käytännössä ratkaisut käyvät mahdottomiksi kun dimensiot kasvavat, enkä nyt tarkoita edes Survo-ristikon rivien ja sarakkeiden määriä vaan niiden mahdollisten vaihtoehtojen määriä.

Edellä kuvatussa ratkaisussa Kroneckerin tulosta muodostui jo lähes 100000 rivin matriisi, ja kun siinä hetkellisesti oli 256 saraketta, se tarvitsi n. 256*10^5*8=204800000, siis n. 200 megatavua levytilaa. Eräässä toisessa, jälkeenpäin kokeilemassani tilanteessa Kroneckerin tulo olisi johtanut matriisiin, jossa olisi ollut yli 10 miljoonaa riviä. Se olisi vienyt laajimmillaan levytilaa yli 20 gigatavua. En kokeillut käytännössä vaan laskin nuo dimensiot etukäteen ja totesin että lienee paras luovuttaa.

Ratkaisutapani toinen rajoitus on sen hitaus: se ei millään pärjää yleisille ratkaisuohjelmille. Toisaalta siinä pääsee näkemään kaikki välivaiheet, mikä on ihan mielenkiintoista sinänsä.