Nacionalinės (įmonių, gyventojų, bankų ir valdžios) skolų tarpusavio užskaitos algoritmas

(rimtos krizės atveju, kaip Didžiosios finansų krizės ar koronaviruso pandemijos)

Raimondas Kuodis

[Versija 1.0 2020.03.02]
[Ši versija 2.1 2020.04.14]

Turinys

PROBLEMOS IŠTAKOS

R.K. pokalbis su LPK prezidentu B.Lubiu 2009 m. krizės pradžioje.

B.L. skundėsi, kad daug įmonių turi apyvartinių lėšų problemą, nors turi daug gautinų sumų iš kitų įmonių. O bankai nebenori skolinti apyvartinėms...

R.K. pasiūlė įmonių skolų užskaitą su žemiau pateikiamu algoritmu. Po kelių metų tai buvo pasiūlyta ŪkMinui, bet tas nematė problemos. Vienas vice ministrų buvo neseniai rimčiau susidomėjęs , bet kol kas reikalai rimčiau nepajudėjo. Gal korona bus postūmis pagaliau?

UŽSKAITOS FILOSOFIJA

a) privačios skolos turi išorinių efektų (externality) – finansinis stabilumas, finansializacija etc.;
b) SVV struktūriškai kenčia nuo to, kad didieji (prekybos centrai etc.), net turėdami lėšų vengia atsiskaityti, finansuoja plėtrą iš to;
c) tas sukuria skolų kaskadas, kurios eksponentiškai sprogsta didelių krizių laikais;
d) iš viso to laimi bankai/faktoringo verslas, nes SVV eina kaulyti apyvartinėms kreditų už didelius procentus ar didelę diskonto normą.

MATEMATINIS MODELIS

Aibės.
i – didžioji užskaitos dalyvių aibė (apimant įmones, bankus, FinMiną[1], gyventojus) aibė. Gerai, kai dalyvautų ir gyventojai, kuriems įmonės ar valdžia įsiskolina algų, bet jie turi būsto paskolų, skolų komunalininkams etc.)

[1] FinMinas krizės metu pats yra problemos šaltinis, ne tik auka – valstybė irgi vengia apmokėti sąskaitas tiekėjams, kai mokestinės įplaukos krenta.

j – *i* aibę dubliuojanti (alias) aibė
if – įmonių poaibis
ib – bankų poaibis
[*ih* – gyventojų poaibis
ig – valdžios institucijų poaibis]

Pradiniai duomenys.
$B_{i,j}$ – pradinė pretenzijų/įsipareigojimų matrica (jos elementas (*i*,*j*) rodo kiek *i* dalyvis paskolinęs *j*)

Kintamieji.
A_i – bendrosios dalyvio *i* pretenzijos
L_j – bendrieji dalyvio *j* įsipareigojimai
N_i – grynosios dalyvio *i* pretenzijos
Z_{ij} – pretenzijų/įsipareigojimų matrica po užskaitos
Obj – tikslo funkcijos kintamasis

Lygtys.
apskaitinės:

$$A0_j = \sum_i B_{ij}$$
$$L0_i = \sum_j B_{ij}$$
$$N0_j = A0_j - L0_j$$
$$Z_{ij} = B_{ij} - X_{ij}$$
$$A_j = \sum_i Z_{ij}$$
$$L_i = \sum_j Z_{ij}$$

$$N_j = A_j - L_j$$

apribojimai:

$$N_j = N0_j$$
$$X_{ij} \geq 0$$

tikslo funkcija:

$$\min Obj = \sum_j \sum_i Z_{ij}$$

UŽSKAITOS MODELIAVIMO REZULTATŲ ILIUSTRACIJA

Pradinė *B* matrica:

	if1	if2	if3	ib1	ib2	ib3	**Liabs**
if1		5	15				**20**
if2	10		25				**35**
if3	20	30					**50**
ib1							**0**
ib2							**0**
ib3							**0**
Assets	**30**	**35**	**40**	**0**	**0**	**0**	105
Liabs	20	35	50	0	0	0	105
A-L	**10**	**0**	**-10**	**0**	**0**	**0**	0

Užskaitos matrica *X*:

	if1	if2	if3	ib1	ib2	ib3
if1	0	5	15	0	0	0
if2	10	0	25	0	0	0
if3	10	30	0	0	0	0
ib1	0	0	0	0	0	0
ib2	0	0	0	0	0	0
ib3	0	0	0	0	0	0

Galutinė *Z* matrica:

	if1	if2	if3	ib1	ib2	ib3	**Liabs**
if1	0	0	0	0	0	0	**0**
if2	0	0	0	0	0	0	**0**
if3	10	0	0	0	0	0	**10**
ib1	0	0	0	0	0	0	**0**
ib2	0	0	0	0	0	0	**0**
ib3	0	0	0	0	0	0	**0**
Assets	**10**	**0**	**0**	**0**	**0**	**0**	10
Liabs	0	0	10	0	0	0	10
A-L	**10**	**0**	**-10**	**0**	**0**	**0**	0

Matome, kad pradinė bendrųjų pretenzijų ekonomikoje suma (€105) modelio buvo smarkiai sumažinta (iki €10), nors grynosios įmonių pretenzijos liko tos pačios (taip ir turi būti, nes čia nėra skolų nurašymo ar perskirstymo modelis).

KĄ DUOTŲ UŽSKAITA

Užskaita būtų vienkartinė (ar periodinė), o po jos reiktų įgyvendint ankstesnį R.K. pasiūlymą ŪkMinui dėl priverstinio ECB MRO palūkanų + 8 p.p. taikymo, tarkim, >2 mėn. trunkančioms prekybinėms skoloms, kad problemos nesikauptų/taptų mažesnės. (Vilkinamų skolų ES direktyva LT buvo perkelta „švelniai“, kas su LT mentalitetu reiškia, kad beveik neperkelta:).
Užskaita gali būti lanksti: apimti ir įmonių pretenzijas bankams (indėlius) (kvadrantas (*ib*, *if*)), bet galima ir be jo – modelis lankstus. Bihevioristinė ekonomika paaiškina, kodėl žmonės/įmonės turi vienu metu ir kredito kortelės skolą ir indėlį banke, kas atrodo neracionalu.

SPRENDIMO TRŪKUMAI IR KIEK JIE RIMTI

Tipinis skepticizmas: skola skolai nelygu, nes skiriasi jos grąžinimo tikimybės pagal dalyvius. O modelis jas užskaito pagal nominalą.
Atsakas: didelės krizės atveju vis tiek skolų voratinklio problema užgrius ekonomiką ir tos nemokumo tikimybės taps net sunkiai įvertinamos – dabar sveika įmonė taps nesveika, jei jos sveiki buvę partneriai taps nesveiki etc.

Be to, valstybei rekomenduočiau laiduoti visą užskaitos procesą, kad dalyviams būtų ramiau.

Verslo viešojo rėmimo priemonių efektyvumas: užskaitos algoritmas leidžia suprastint problemą nuo *gross* iki *net* lygio. *Gross* filosofija gerokai brangesnė nei *net*.

Pavyzdžiui, LT, Europa etc. paleidžia VP pirkimo programas. Bet kas iš to? Jei kokie UAB „Seniukai“ gaus mainais į pretenzijas kitoms įmonėms gaus per valdžios programą likvidžias lėšas (*cash*‘ą), tai NEGARANTUOJA, kad jis bus nuleistas žemyn (atsiskaityta su tiekėjais, darbuotojais etc.), o ne pvz. išsimokėtos premijos (ypač jaučiant, kad įmonė bankrutuos).

Antra, šios programos pirks VP, o šių SVV paprastai net neleidžia – taigi, jos yra stambiam verslui skirtos programos!
Užskaita gi išvalo skolos grandines iki jų galo, cash‘o jai nereikia, o su *net* pasekmėm gerokai lengviau/pigiau valdžiai tvarkytis nei su *gross*.

Galiausiai, nežinom ar *gross* procesui valdžia turės pakankamai finansinių resursų.

Išvada: tad geriau taikyti netobulą užskaitos procesą, negu sulaukt „teisiškai nepriekaištingų“ skolų piramidės griūties ar šaudyt į tą piramidę iš pipetės (turint mažus išteklius.

MODELIO KODAS

Modelio kodas yra GAMS (General Algebraic Modelling System) kalba ir paruoštas naudoti (*debuggintas, kompiliuojasi*)[2]:

```
*### 1. AIBIU DEKLARAVIMAS ###
Sets

i dalyviai
/123
 125
 127/

*/
*$include rk_netting_set.txt
*/
;

Alias (i,j);

*### 2. PRADINIAI DUOMENYS ###

Parameter B(i,j);
*$include rk_netting_data5.txt
*;

B("123","125")=5.15;
B("123","127")=15.20;
```

[2] Žvaigždute (`*`) prasidedanti kodo eilutė (ar `$ontext $offtext` komandom įrėmintos dalys) GAMS sintaksėje reiškia... komentarą. Taip kode galima palikti, pavyzdžiui, duomenų importo alternatyvas, kaip aptarta žemiau. Jei renkatės vieną jų, tai ją „atžvaigždukinkit“, o kitą alternatyvą „užžvaigždukinkit“.

```
B("125","123")=10.30;
B("125","127")=25.17;
B("127","123")=20.00;
B("127","125")=30.00;

$ontext
Table B(i,j)
    i1    i2    i3    i4    i5    i6    i7    i8    i9   i10
i1         7     8     3     2     3     6     9     6     8
i2   5           1     2     3     4     5     5     3     3
i3   4     9           2     6     8     4     7     3     7
i4   9     8     2           9     8     8     3     4     6
i5   7     7     3     2           1     4     6     7     7
i6   5     6     5     1     5           6     5     1     5
i7   3     5     6     8     3     4           5     9     5
i8   1     4     6     2     6     7     2           0     4
i9   0     3     4     3     5     2     0     6           5
i10  1     2     2     4     5     0     9     7     6
;
$offtext

*### 3. KINTAMIEJI ###
Variables

A0(j)
L0(i)
N0(j)
Obj0

X(i,j)
Z(i,j)
```

```
A(j)
L(i)
N(j)

Obj;

*### 4. LYGTYS ###
Equations

eq_A0(j)
eq_L0(i)
eq_N0(j)
eq_Obj0

eq_Z(i,j)

eq_A(j)
eq_L(i)
eq_N(j)

eq_NN(j)

eq_Obj;

*---Pradiniai A,L,N---
eq_A0(j)..    A0(j) =e= sum(i, B(i,j));
eq_L0(i)..    L0(i) =e= sum(j, B(i,j));
eq_N0(j)..    N0(j) =e= A0(j) - L0(j);
eq_Obj0..     Obj0 =e= sum((i,j), B(i,j));

*---Nettingas---
eq_Z(i,j)..  Z(i,j) =e= B(i,j) - X(i,j);
```

```
*---Galutiniai A,L,N---
eq_A(j)..     A(j) =e= sum(i, Z(i,j));
eq_L(i)..     L(i) =e= sum(j, Z(i,j));
eq_N(j)..     N(j) =e= A(j) - L(j);

*---Apribojimas---
eq_NN(j)..    N(j) =e= N0(j);

*---Tikslo funkcija---
eq_Obj..      Obj =e= sum((i,j), Z(i,j));

*### 5. RIBOS/PRADINES_REIKSMES ###
*---Orientacines reiksmes---
*X.l(i,j) = 0;
*--- Apribojimai---
Z.lo(i,j) = 0;
X.up(i,j) = 1000;

*### 6. MODELIO DEKLARAVIMAS/SPRENDIMAS ###
Model rk_netting /all/;

*Z.l(i,j) = 0$(not B(i,j));

Solve rk_netting USING lp MINIMIZING Obj;

*### 7. SPRENDINIO ISVEDIMAS ###
File sol /'z_out.txt'/;
Put sol;
sol.pc=6;

put 'Obj: Obj0(i), Obj'; put/;
```

```
put Obj0.l; put Obj.l; put/;

put 'ALN: A0(i), L0(i), N0(i), A(i), L(i), A(i)'; put/;
loop(i, put i.tl; put A0.l(i):6:3; put L0.l(i):6:3; put N0.l(i):6:1;
                  put A.l(i):6:3; put L.l(i):6:3; put N.l(i):6:3; put/);

*=== Z by a/l ===
put '*=== Z pagal L(liabs) ==='; put/;
loop(i, put i.tl put/; loop(j, put j.tl; put Z.l(i,j):6:3 put/); put/);

put '*=== Z pagal A(assets) ==='; put/;
loop(j, put j.tl put/; loop(i, put i.tl; put Z.l(i,j):6:3 put/); put/);

*put 'Z: Z(i,j)'; put/;
*loop(i, put i.tl; loop(j, put Z.l(i,j):6:3); put/);
```

KODO KOMENTARAS IR NAUDINGI PATARIMAI MODELIUOTOJUI (D.U.K.)

```
*### 1. AIBIU DEKLARAVIMAS ###

Sets

i dalyviai
/
37108011234
18812341234
/
```

Čia turi atsirasti unikalūs dalyvių identifikatoriai (pavyzdžiui, darbuotojo asmens kodas (37108011234), įmonės kodas (18812341234) .

Didelės užskaitos atveju geriausiai tą padaryti importuojant struktūrizuotą duomenų failą. Tas (Excel, csv, txt) failas turės stulpelį su aibe *i* – įmonių, žmonių kodais, kurie dalyvauja užskaitoj (kodų tvarka nesvarbi):
18812341234
37108011234
ir t.t...

	A
1	37108011234
2	18812341234
3	

```
/
$include rk_netting_set.txt
/
;
```

Dubliuojanti *i* aibę *j* aibė sukuriama taip:

```
Alias (i,j);
```

```
*### 2. PRADINIAI DUOMENYS ###
```

Deklaravę pradinę balansų matricą...

```
Parameter B(i,j);
```

Užpildome ją duomenim. Pirmas variantas su išorinio failo importu jau matytas...

```
*$include rk_netting_data5.txt
*;
```

Importas turi įrašyti į GAMS kodo failą tokią informaciją „stulpelio“ formatu (kas reiškia, kad dalyvis su kodu 123 skolingas dalyviui 125 5.15€). Tai patogiausias būdas didelės užskaitos atveju...

```
B("123","125")=5.15;
B("123","127")=15.20;
B("125","123")=10.30;
B("125","127")=25.17;
B("127","123")=20.00;
B("127","125")=30.00;
```

... arba, mažos užskaitos atveju, tiesiog galima suformuoti matricą.

```
Table B(i,j)
    i1    i2    i3    i4    i5    i6    i7    i8    i9   i10
```

```
i1          7       8       3       2       3       6       9       6       8
i2    5             1       2       3       4       5       5       3       3
i3    4       9             2       6       8       4       7       3       7
i4    9       8       2             9       8       8       3       4       6
i5    7       7       3       2             1       4       6       7       7
i6    5       6       5       1       5             6       5       1       5
i7    3       5       6       8       3       4             5       9       5
i8    1       4       6       2       6       7       2             0       4
i9    0       3       4       3       5       2       0       6             5
i10   1       2       2       4       5       0       9       7       6
;
```

... arba, mažos užskaitos atveju, tiesiog galima suformuoti matricą.

Kas yra „gera užskaitai“ pradinė B(*i*,*j*) matrica?

- turi būti „pilna“ potencialiai išnaikintinų skolų grandinių. Tam reikia, kad dalyvautų kuo daugiau dalyvių...
- ... bet neturėtų apimti „smulkių“ duomenų, pavyzdžiui, kažkas kažkam skolingas 1€. Todėl geriau uždėti minimalią sumą elementui (*i*,*j*).

```
*### 3-4.
```

Nekontraversiška.

```
*### 5. RIBOS/PRADINES_REIKSMES ###
```

Gera praktika yra „užvesti“ modelį ant sprendinio per orientacines reikšmes...

```
*---Orientacines reiksmes---
*X.l(i,j) = 0;
```

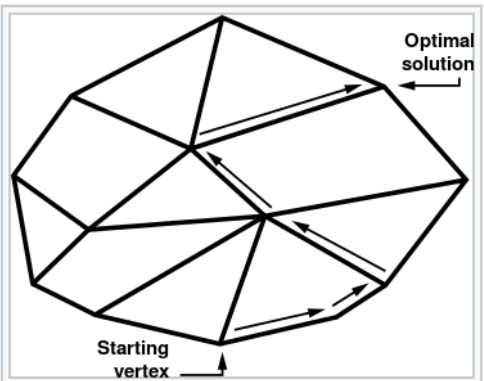
... ir/ar uždedant apribojimus kintamiesiems, kad optimizatorius (solver) nekeliautų į plačias lankas sprendinio beieškodamas.

```
*--- Apribojimai---
Z.lo(i,j) = 0;
X.up(i,j) = 1000;


*### 6. MODELIO DEKLARAVIMAS/SPRENDIMAS ###

Model rk_netting /all/;
Solve rk_netting USING lp MINIMIZING Obj;
```

`lp` reiškia tiesinį programavimą (linear programming). Valio, tas didina tikimybę, kad modelis „išsispręs“. Bent jau 10x10 pradinę matricą redukavo per sekundę ant personalinio kompiuterio. Žinoma, jei matrica bus 1 000 000x1 000 000, visko gali nutikt. „Sprogstamosios kombinatorikos“ geriau vengt, nes pradinė pretenzijų/įsipareigojimų matrica turi $n(n-1)$ elementą. Bet kadangi čia `lp`, o solver'is yra elementarus simplex algo [https://en.wikipedia.org/wiki/Simplex_algorithm], tai greičiausiai sprendinio radimas tėra (kaip kompiuterių mokslininkai sako) brutalios jėgos (brutal force) reikalas, kaip parodyta paveiksle.



A system of linear inequalities defines a polytope as a feasible region. The simplex algorithm begins at a starting vertex and moves along the edges of the polytope until it reaches the vertex of the optimal solution.

Solver'į šitam projekte naudoju Stanford University (Systems Optimization Laboratory) MINOS 5.6
https://en.wikipedia.org/wiki/MINOS_(optimization_software)

Bet yra ir kitų, kaip CPLEX etc., kurie daro iš esmės tą patį.

Kitos algebrinio modeliavimo platformos (Matlab etc.) irgi juos „išsikviečia" modelio sprendimui, tad nėra didelio skirtumo ar šį projektą darysit su GAMS ar ne☺

Sėkmės! Jei turit klausimų/pastebėjimų/pasiūlymų: rkuodis@lb.lt, www.facebook.com/raimondas.kuodis

*** Pabaiga***