1

Modellering

Syfte

att visa några typiska delar som finns i matematiska modeller, och hur man modellerar dem matematiskt och i modelleringsspråk.
Vi använder AMPL (A Mathematical Programming Language) då det har någorlunda lättbegriplig syntax. Andra exempel är Lindo, MPL, GAMS e.t.c.

2

Lagerhållning

Vi har ett lager. Under en tidsperiod sker både tillverkning/införskaffning samt försäljning.
Antag att vi har endast en vara och en period.

Med definitioner

$t = 0, 1$ tidsperiod.

x_t lagerstatus.

y_t inköp, $t = 1$.

z_t försäljning $t = 1$.

får vi formuleringen:

$$x_1 = x_0 + y_1 - z_1$$
$$x, y, z \geq 0$$

3

Lager, varianter

Med flera perioder får vi

T Sista tidsperioden

$t \in 0 \dots T$ tidsperioder

x_t varulager

y_t inköp

z_t försäljning

c ursprungligt lager.

med formulering:

$$x_t = x_{t-1} + y_t - z_t, \quad \forall t \in 1 \dots T$$
$$x_0 = c + y_0 - z_0$$
$$x, y, z \geq 0$$

4

Lager, varianter

Vi kan i stället tänka oss en situation där vi mäter värdeutvecklingen på en tillgång (t.ex en aktie). Vi låter nu x representera *värdet* av våra aktier. Vi behöver en extra parameter ρ_t som representerar värdeutvecklingen hos tillgången mellan period $t-1$ och t. I fallet aktier är ρ det samma som priset vid tid t delat med priset vid tid $t-1$ Vi får

$$x_t = \rho x_{t-1} + y_t - z_t, \quad \forall t \in 1 \dots T$$
$$x_0 = c + y_0 - z_0$$
$$x, y, z \geq 0$$

5

Portföljproblem

Vi antar vidare att vi har ett antal olika tillgångar att investera våra tillgångar i. Våra variabler får nu ett extra index i för tillgångskategorin .

I Tillgångskategori

x_i^t Tillgångar av kategori i vid tod t.

y_i^t inköp

z_i^t försäljning

c_i ursprungliga tillgångar.

$$x_i^t = \rho_i^t x_i^{t-1} + y_i^t - z_i^t, \quad \forall t \in 1 \dots T, \forall i \in I$$
$$x_i^0 = c_i + y_i^0 - z_i^0 \quad , \forall i \in I$$
$$x, y, z \geq 0$$

6

Portföljproblem

Vi antar vidare att pengar flödat in och ut ur vår portfölj med summan g_t per tidsperiod. I övrigt måste köp och försäljning av tillgångar vara självfinansierade. Dessutom önskar vi maximera tillgångarna vid periodens slut.

$$\max \sum_{i \in I} x_i^T$$
$$\text{då} \sum_{i \in I} (y_i^t - z_i^t) = g_t, \forall t \in 0 \dots T$$
$$x_i^t = \rho_i^t x_i^{t-1} + y_i^t - z_i^t, \quad \forall t \in 1 \dots T, \forall i \in I$$
$$x_i^0 = c_i + y_i^0 - z_i^0 \quad , \forall i \in I$$
$$x, y, z \geq 0$$

7

Transaktionskostnader

Vidare kan vi införa transaktionskostnader genom att modifiera det första villkoret från

$$\sum_{i \in I} (y_i^t - z_i^t) = g_t, \quad \forall t \in 0 \dots T$$

till

$$\sum_{i \in I} (y_i^t (1 - \gamma) - (1 + \gamma) z_i^t) = g_t, \quad \forall t \in 0 \dots T$$

D.v.s. om vi säljer något värt y_i får vi bara ut $(1-\gamma)y_i$ kronor. Om vi köper något värt z_i kronor måste vi betala $(1+\gamma)z_i$ kronor.

8

Modellfil

```
param T;

set TIMESTEPS ordered :=0..T;
set ENDSTEPS ordered := TIMESTEPS diff {first(TIMESTEPS)};
set ASSETS;

param asset_developpment{ENDSTEPS,ASSETS};
param initial_assets{ASSETS};
param inflow{TIMESTEPS};

var owned{TIMESTEPS,ASSETS} >=0;
var buy{TIMESTEPS,ASSETS} >=0;
var sell{TIMESTEPS,ASSETS} >=0;
```

```
maximize end_wealth:
         sum{ a in ASSETS} owned[last(TIMESTEPS),a];

subject to mass_conservation{t in TIMESTEPS}:
     sum{ a in ASSETS}(buy[t,a]-sell[t,a])=inflow[t];

subject to developpment{t in TIMESTEPS, a in ASSETS:
                               t != first(TIMESTEPS)}:
     owned[t,a]=asset_developpment[t,a]*owned[prev(t),a]
          +buy[t,a]-sell[t,a];

subject to initial_owned{a in ASSETS}:
     owned[first(TIMESTEPS),a]=initial_assets[a]
       +buy[first(TIMESTEPS),a]-sell[first(TIMESTEPS),a];
```


Datafil

```
param T := 3;

set ASSETS := stock bond;

param asset_developpment:
         stock    bond :=
1        0.97     1.03
2        1.05     1.05
3        1.2      1.04;
```



```
param  initial_assets :=
stock 1
bond 1;

param  inflow:=
0 0
1 0.1
2 0.5
3 0.2;
```


Mängder

I AMPL definieras mängder med kommandot `set`. Formellt har vi

set <namn> [attribut] [:= definition] ;

(<argument> betyder att argumentet skall vara med, [argument] att det kan vara med)

Attribut kan vara (Bl.a.)

```
ordered
within <mängd>
circular
```

Ordered betyder att mängden är ordnad, och vi kan använda kommandon som first, last, prev, next, member

Within betyder att vi har definierat en delmängd.

Mängdoperationer

Vi kan dessutom ha operationer på mängder som

union	union $A \cup B$
diff	Mängddifferans $A \setminus B$
symdiff	Symetrisk mängddifferans $((A \setminus B) \cup (B \setminus A))$
inter	Snitt $A \cap B$
cross	Produkt $A \times B$.

Parametrar

prametrar definieras med kommandot param. Formellt har vi

`parm <namn> [indexering] [attribut] [:= definition ] ;`

de vanligare attributen är

```
default <expr>
>=,=,<=,!= <expr>
```

Attributet default ger ett värde som används om vi ej tilldelar parametern något värde

>= <expr> används som kontroll. Om vi t.ex deklarerar

```
param kalle default 1, >=0,<=2, !=1.3;
```

får vi en parmeter vars värde blir 1 om vi ej senare tilldelar den ett värde och vi kan ej ge den ett värde utanför intervallet [0,2].

Variabler

Variabler definieras med `var` Formellt har vi

`var <namn> [indexering] [attribut] ;`

de vanligare attributen är

```
integer
binary
>=,=,<= <expr>
```

Om vi till exemple deklarar

```
set MITT_SET;
param kalle_max{MITT_SET};
var kalle{i in MITT_SET} integer , >=0, <= kalle_max[i];
```

får vi en variabel kalle som måste vara heltalig, vars lägsta värde är 0 och vars högsta värde ges av parametern kalle_max;

Målfunktion

Målfunktion deklareras med `maximize` eller `minimize` beroende på riktning. Deklarationen blir

```
maximize <namn>: <expr>;
```

där <expr> står för ett godtyckligt uttryck i våra variabler och konstanter.

17

Bivillkor

Bivlllkor deklareras med `subject to`

Formellt har vi

```
subject to <namn> [indexering]: <expr> <op> <expr>;
```

<op> är >=, = eller <= .

18

Uttryck

Uttryck är godtyckliga formler i våra variabler och konstanter. Några funktioner som kan vara bra är summor, produkter, max och min Vi tar ett bivillkor som ett exempel;

```
set MINA_TAL:=1..10;
param knatte{MINA_TAL};
param fnatte{MINA_TAL};
var tjatte{MINA_TAL};

subject to mitt_biv_1:
sum {i in MINA_TAL,j in MINA_TAL}knatte[i]*fnatte[j]>=
         prod {i in MINA_TAL} tjatte[i];

subject to mitt_biv_2 { k in MINA_TAL}:
max { i in MINA_TAL} knatte[i] >=  tjatte[k];
```

19

Indexering

Formellt har vi

```
{<variabel> in <set>[, ..., <variabel> in <set>]
[:<expr>]}
```

Exempel:

```
var kalle{1..10,1..10};

subject to biv {i in 1..10, j in 1..10: j<i}:
                         kalle[i,j]>=2;
```

20

Scenarioträd

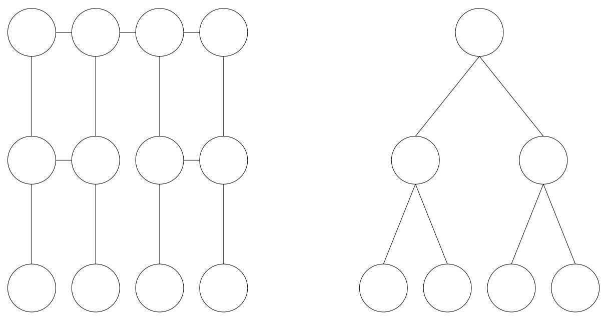
Om vi tittar på vårt portföljproblem ser vi att vi än så länge saknar slump i problemet.

När vi lägger till scenarier kan vi göra detta på två olika sätt, på scenario eller trädform, vilket svarar mot med eller utan "non-anticipativity constraints".

I trädform ser vi till att variabler i tidiga steg är oberoende av framtiden genom att endast ha en kopia. I scenario form ser vi till att de är oberoende av framtiden med explicita bivillkor.

21

scenario och trädform



22

Portföljproblem

Lägger vi till ett index för scenario så får vi

$$\max \sum_{i\in I, s\in S} p(s) x^T_{i,s}$$
$$\text{då} \sum_{i\in I}(y^t_{i,s} - z^t_{i,s}) = g^t_s, \forall t \in 1\ldots T, \forall s\in S$$
$$x^t_{i,s} = \rho^t_{i,s}x^{t-1}_{i,s} + y^t_{i,s} - z^t_{i,s}, \quad \forall t\in 1\ldots T, \forall i \in I, \forall s \in S$$
$$x^0_{i,s} = c_i \quad, \forall i\in I, \forall s\in S$$
$$x, y, z \geq 0$$

23

Scenarioformulering

Vi kan inte längre garantera att $x^t_{i,s}$ är oberoende av information som vi ej känner vid tid t. Vi inför non-anticipativity-bivillkor

$$x^t_{i,s1} = x^t_{i,s2} \quad \forall i\in I, \forall (t, s1, s2) \in Q$$
$$y^t_{i,s1} = y^t_{i,s2} \quad \forall i\in I, \forall (t, s1, s2) \in Q$$
$$z^t_{i,s1} = z^t_{i,s2} \quad \forall i\in I, \forall (t, s1, s2) \in Q$$

med Q definierat av

$$Q = \{(1,1,2),(1,2,3),(1,3,4),(2,1,2),(2,3,4)\}$$

24

nodformulering

Vi kan ersätta både indexen s och i med information som talar om i vilken nod vi befinner oss. Samtidigt introducerar vi några olika mängder och en operator

N	Mängden av noder.
L	Mängden av löv.
R	Mängden av rot-noder (Bara en).
$p(n) : N\setminus R \to N$	operator som pekar ut föräldern till en nod.

Notera: noden ger information *både* om scenario och tidssteg.

Vår problem blir då

$$\max \sum_{i \in I, n \in L} x_i^n p(n)$$
$$\text{då} \sum_{i \in I} (y_i^n - z_i^n) = g^n, \forall n \in N$$
$$x_i^n = \rho_i^n x_i^{p(n)} + y_i^n - z_i^n, \quad \forall i \in I, \forall n \in N \setminus R$$
$$x_i^n = c_i + y_i^n - z_i^n \quad, \forall i \in I, \forall n \in R$$
$$x, y, z \geq 0$$

AMPL Modellfil

```
param last_node;
set NODES ordered := 0..last_node;

param prob{NODES};
param parent{NODES};
param level{n in NODES}:=if (parent[n]==-1)
                          then 0
                          else (1+level[parent[n]]);

set LEAFS within NODES := setof{ n in NODES :
                 level[n]=max{i in NODES} level[i]} n;

set ROOT within NODES := setof{ n in NODES :
                 parent[n]==-1} n;
```



```
set ASSETS;

param asset_developpment{ASSETS,NODES diff ROOT};
param initial_assets{ASSETS};
param inflow{NODES};


var owned{ASSETS,NODES} >=0;
var buy{ASSETS,NODES} >=0;
var sell{ASSETS, NODES} >=0;
```



```
maximize end_wealth:
     sum{ a in ASSETS, n in LEAFS} owned[a,n]*prob[n];

subject to mass_conservation{n in NODES}:
     sum{ a in ASSETS}(buy[a,n]-sell[a,n])=inflow[n];

subject to developpment{n in NODES diff ROOT, a in ASSETS}:
     owned[a,n]=
          asset_developpment[a,n]*owned[a,parent[n]]
          +buy[a,n]-sell[a,n];

subject to initial_owned{a in ASSETS, n in ROOT}:
     owned[a,n]=initial_assets[a]+buy[a,n]-sell[a,n];
```


Datafil

```
param last_node := 6;
param :
   parent prob:=
0    -1    1
1    0     0.5
2    0     0.5
3    1     0.25
4    1     0.25
5    2     0.25
6    2     0.25;
```



```
set ASSETS := stock bond;

param asset_developpment (tr) :
         stock    bond :=
1        1.00     1.03
2        1.2      1.04
3        0.9      1.02
4        1.05     1.05
5        1.1      1.04
6        1.0      1.04;
```



```
param  initial_assets :=
stock 1
bond 1;

param  inflow:=
0 0
1 0.1
2 0.5
3 0.2
4 -0.2
5 -0.1
6 0.1;
```


Symmetriska träd

Om vi vill slippa ange hela trädet för hand kan vi skapa symmetriska träd automatiskt.

```
param levels;
param split;
param last_node:=sum{i in 1..levels-1} split^i;
set NODES ordered := 0..last_node;

param parent{n in NODES} := if n=0 then -1
                                   else (n-1) div split;
param child_no{n in NODES} := if n=0 then -1
                                     else (n-1) mod split;
param level{n in NODES}:=if parent[n]==-1 then 0
                                  else 1+level[parent[n]];
```

```
set LEAFS within NODES :=
setof{ n in NODES :level[n]=max{i in NODES} level[i]} n;

set ROOT within NODES :=
setof{ n in NODES : parent[n]==-1} n;

param pre_prob{0..split-1};
param prob{ n in NODES} :=
if n == 0 then 1 else prob[parent[n]]*pre_prob[child_no[n]];

set ASSETS;

param asset_developpment{ASSETS,0..split-1};
param initial_assets{ASSETS};
param inflow{0..split-1};
```



```
param inflow_root;

var owned{ASSETS,NODES} >=0;
var buy{ASSETS,NODES} >=0;
var sell{ASSETS, NODES} >=0;

maximize end_wealth:
     sum{ a in ASSETS, n in LEAFS} prob[n]*owned[a,n];

subject to mass_conservation{n in NODES diff ROOT}:
  sum{ a in ASSETS}(buy[a,n]-sell[a,n])=inflow[child_no[n]];

subject to mass_conservation_root{n in ROOT}:
     sum{ a in ASSETS}(buy[a,n]-sell[a,n])=inflow_root;
```



```
subject to developpment{n in NODES diff ROOT, a in ASSETS}:
     owned[a,n]=
       asset_developpment[a,child_no[n]]*owned[a,parent[n]]
       +buy[a,n]-sell[a,n];

subject to initial_owned{a in ASSETS, n in ROOT}:
     owned[a,n]=initial_assets[a]+buy[a,n]-sell[a,n];
```


Mer problem: blandningsproblem

Ett klassiskt problem är blandnigsproblemet. Vi har en mängd ingredienser I vilka innehåller olika mängder av olika substanser S. Vi skall blanda dessa ingredienser så att vi får ett antal olika produkter P vilka har övre och undre gränser på tillåtna värden för de olika substanserna S. De olika ingredienserna kostar olika mycket och kan dessutom finnas i begränsade mängder.

Matematisk formulering

$i \in I$ Index för ingrediens.

$p \in P$ Index för produkt.

$s \in S$ Index för substans.

a_{is} Andel av ingrediens i som består av substans S.

l_{ps}, u_{ps} Övre och undre gränser för andel av substanser i produkter

c_s Kostnad för ingrediens i

m_i Max total mängd av ingrediens i.

g_p Önskad mängd av produkt p.

x_{ip} Mängd ingrediens i i produkt p.

Matematisk formulering:2

$$
\begin{array}{lll}
\min & \sum_{i\in I} c_i \sum_{p\in P} x_{ip} & \\
\text{då} & \sum_{i\in I} x_{ip} & = g_p \quad \forall p \in P \\
& \sum_{i\in I} a_{is} x_{ip} & \geq l_{ps} g_p \quad \forall p \in P, s \in S \\
& \sum_{i\in I} a_{is} x_{ip} & \leq u_{ps} g_p \quad \forall p \in P, s \in S \\
& \sum_{p\in P} x_{ip} & \leq m_i \quad \forall i \in I \\
& x_{ip} & \geq 0 \quad \forall i \in I, p \in P
\end{array}
$$


```
set INGREDIENS;
set PRODUKT;
set SUBSTANS;

param andel{INGREDIENS,SUBSTANS} >=0;
param undre_gräns{PRODUKT,SUBSTANS} >=0;
param övre_gräns{PRODUKT,SUBSTANS} >=0;
param kostnad{INGREDIENS};
param max_total{INGREDIENS};
param önskad_mängd{PRODUKT};

var x{INGREDIENS,PRODUKT}>=0;
```



```
minimize total_kostnad:
     sum{i in INGREDIENS, p in PRODUKT}x[i,p]*kostnad[i];

subject to efterfrågan {p in PRODUKT}:
     sum{ i in INGREDIENS} x[i,p] = önskad_mängd[p];

subject to undre_andel{s in SUBSTANS, p in PRODUKT}:
     sum{ i in INGREDIENS} x[i,p]*andel[i,s] >=
     undre_gräns[p,s]*önskad_mängd[p];

subject to övre_andel{s in SUBSTANS, p in PRODUKT}:
     sum{ i in INGREDIENS} x[i,p]*andel[i,s] <=
     övre_gräns[p,s]*önskad_mängd[p];
subject to max_lager{i in INGREDIENS}:
     sum{p in PRODUKT} x[i,p] <= max_total[i];
```

41

Fasta kostnader

Ibland behöver man modellera fasta kostnader, t.ex att bygga en fabrik, vilket kostar pengar redan innan vi producerar något. Vi kan titta på vårt ursprungliga protförljproblem, och anta att förutom en proportionell transaktionskostnad γ har vi en avgift a vilken uppkommer oavsett hur lite vi köper eller säljer. Vi får då lägga till följande saker i problemdefinitionen

a Fast kostnad.

q_i^t Binär indikatorvariabel.

M Ett mycket stort tal.

42

Fasta kostnader

Vi får ändra vår modell som följer:

$$\begin{aligned} \max \quad & \sum_{i\in I} x_i^T - \sum_{i\in I, t\in T} a q_i^t \\ \text{då} \quad & z_i^t + y_i^t \le M q_i^t \quad \forall i \in I, t \in 1\ldots T \end{aligned}$$

i AMPL:

```
param fast_kostnad >=0;
var indikator{ENDSTEPS,ASSETS} binary;
param M;
```

43

Extraresurser

Ett vanligt förekommande element i modellering är att vi har extra resurser tillgängliga till ett högre pris. Om vi tittar på vårt blandningsproblem, där vi hade en maxkapacitet på hur mycket som kunde användas av varje ingrediens, kan vi anta att vi kan köpa extra matrial till ett dyrare pris.

$\bar{c}_i$ Kostnad för extra inköp

y_i Extrainköpt mängd.

44

$$\begin{aligned} \min \quad & \textstyle\sum_{i\in I} c_i \sum_{p\in P} x_{ip} + \sum_{i\in I} (\bar{c}_i - c_i) y_i \\ \text{då} \quad & \textstyle\sum_{i\in I} x_{ip} = g_p \quad \forall p \in P \\ & \textstyle\sum_{i\in I} a_{is} x_{ip} \ge l_{ps} g_p \quad \forall p \in P, s \in S \\ & \textstyle\sum_{i\in I} a_{is} x_{ip} \le u_{ps} g_p \quad \forall p \in P, s \in S \\ & \textstyle\sum_{p\in P} x_{ip} \le m_i + y_i \quad \forall i \in I \\ & x_{ip} \ge 0 \quad \forall i \in I, p \in P \\ & y_i \ge 0 \quad \forall i \in I \end{aligned}$$

45

Flödesproblem (Min cost flow)

Ett annat vanligt problem är min-cost flow. Vi skall nu skicka ett flöde (paket, varor , passagerare) genom ett nätverk. till billigaste kostnad. I vårt nätverk har vi ett antal noder N som förbinds av en mängd av bågar B.

46

Matematisk formulering

$n \in N$	index för nod.
$b \in B \subset N \times N$	index för bågar.
$s^+(b)$	Mängd av bågar ut från b
$s^-(b)$	Mängd av bågar in i b
c_b	Kostnad för att transportera saker längs en båge
k_b	Max transportkapacitet hos en båge.
p_n	Produktion eller konsumtion i en nod.
x_b	Flöde på båge B

47

Matematisk formulering

$$\begin{aligned} \min \quad & \sum_{b\in B} x_b c_b \\ \text{då} \quad & \sum_{b\in s^+(n)} x_b - \sum_{b\in s^-(n)} x_b = p_n \quad && \forall n \in N \\ & 0 \le x_b \le k_b && \forall b \in B \end{aligned}$$

48

```
set NODER;
set BÅGAR within NODER cross NODER;

param max_flöde{BÅGAR}>=0;
param kostnad{BÅGAR};
param produktion{NODER} ;

var x{(s,d) in BÅGAR} >=0 , <= max_flöde[s,d];

minimize total_kostnad:
     sum{(s,d) in BÅGAR} x[s,d]*kostnad[s,d];

subject to massbalans {n in NODER}:
     sum{d in NODER: (n,d) in BÅGAR} x[n,d] -
     sum{s in NODER: (s,n) in BÅGAR} x[s,n] = produktion[n];
```