AMPL-Modell für beliebige Produkte (myprod.mod)

```
set P;

param a {j in P};
param beta;
param c {j in P};
param u {j in P};

var X {j in P};

maximize Profit: sum {j in P} c[j] * X[j];

subject to Zeit:
    sum {j in P} (1/a[j]) * X[j] <= beta;

subject to Grenzen {j in P}: 0 <= X[j] <= u[j];
```

AMPL-Daten Datei (myprod.dat) für myprod.mod

```
set P := Baender Rollen;

param:      a     c    u    :=
  Baender  200   20   6000
  Rollen   140   30   4000 ;

param beta := 40;
```

Lösen in AMPL

```
ampl: model myprod.mod;
ampl: data myprod.dat;
ampl: solve;
MINOS 5.51: optimal solution found.
2 iterations, objective 165714.2857
ampl: display X;
X [*] :=
Baender  2285.71
 Rollen  4000
;
```

1.2 Beispiel: Transportproblem

Situation:

- Von einigen Orten (*origins* O) muss Öl an andere (*destinations* D) transportiert werden.
- An origin $i \in O$ sind s_i Tonnen verfügbar (*supply*), an destination $j \in D$ werden d_j Tonnen benötigt (*demand*).
- Der Transport von i nach j kostet c_{ij} Euro pro Tonne.
- Aufgabe: Finde einen Transportplan, der die Kosten minimiert.

[5]

AMPL-Modell (`transp.mod`)

```
set ORIG;   # origins
set DEST;   # destinations

param supply {ORIG} >= 0;
  # amounts available at origins
param demand {DEST} >= 0;
  # amounts required at destinations
check: sum {i in ORIG} supply[i]
        = sum {j in DEST} demand[j];
param cost {ORIG,DEST} >= 0;
  # shipment costs per unit

var X {ORIG,DEST} >= 0;
  # units to be shipped

minimize Total_Cost:
   sum {i in ORIG, j in DEST} cost[i,j] * X[i,j];

subject to Supply {i in ORIG}:
   sum {j in DEST} X[i,j] = supply[i];

subject to Demand {j in DEST}:
   sum {i in ORIG} X[i,j] = demand[j];
```

AMPL-Daten (`transp.dat`)

```
# define set "ORIG" and param "supply"
param: ORIG:  supply :=
        GARY   1400
        CLEV   2600
        PITT   2900 ;

# define "DEST" and "demand"
param: DEST:  demand :=
        FRA     900
        DET    1200
        LAN     600
        WIN     400
        STL    1700
        FRE    1100
        LAF    1000 ;

param cost:
          FRA  DET  LAN  WIN  STL  FRE  LAF :=
   GARY    39   14   11   14   16   82    8
   CLEV    27    9   12    9   26   95   17
   PITT    24   14   17   13   28   99   20 ;
```

Lösen mit AMPL

```
ampl: model transp.mod;
ampl: data transp.dat;
ampl: solve;
MINOS 5.51: optimal solution found.
13 iterations, objective 196200
ampl: display X;
Trans [*,*] (tr)
:     CLEV   GARY   PITT    :=
DET   1200      0      0
FRA      0      0    900
FRE      0   1100      0
LAF    400    300    300
LAN    600      0      0
STL      0      0   1700
WIN    400      0      0
;
```

Modifikation der Situation:

- Zusätzlich entstehen Festkosten f_{ij}, falls Öl von origin i zu destination j transportiert wird.
- Es dürfen höchstens u_{ij} Tonnen Öl von origin i zu destination j transportiert werden.

AMPL-Modell (`fixtransp.mod`)

```
set ORIG;
set DEST;
param supply {ORIG} >= 0;
param demand {DEST} >= 0;
param cost {ORIG,DEST} >= 0;
  # shipment costs per unit
param fcost {ORIG,DEST} >= 0;
  # fixed costs for starting shipping
param limit {ORIG,DEST} >= 0;
  # upper bounds on units to be shipped

var X {ORIG,DEST} >= 0;
  # units to be shipped
var Y {ORIG,DEST} binary ;
  # indicator variables for shipping

minimize Total_Cost:
   sum {i in ORIG, j in DEST} cost[i,j] * X[i,j]
  +sum {i in ORIG, j in DEST} fcost[i,j] * Y[i,j];

subject to Supply {i in ORIG}:
   sum {j in DEST} X[i,j] = supply[i];
subject to Demand {j in DEST}:
   sum {i in ORIG} X[i,j] = demand[j];
subject to Bound {i in ORIG, j in DEST}:
   X[i,j] <= limit[i,j] * Y[i,j];
```

AMPL-Daten (`fixtransp.dat`)

```
param: ORIG:  supply :=
        GARY   1400
        CLEV   2600
        PITT   2900 ;

param: DEST:  demand :=
        FRA     900
        DET    1200
        LAN     600
        WIN     400
        STL    1700
        FRE    1100
        LAF    1000 ;

param cost:
          FRA  DET  LAN  WIN  STL  FRE  LAF :=
   GARY    39   14   11   14   16   82    8
   CLEV    27    9   12    9   26   95   17
   PITT    24   14   17   13   28   99   20 ;

param fcost:
          FRA  DET  LAN  WIN  STL  FRE  LAF :=
   GARY 3000 1200 1200 1200 2500 3500 2500
   CLEV 2000 1000 1500 1200 2500 3000 2200
   PITT 2000 1200 1500 1500 2500 3500 2200 ;

param limit default 625 ;
```

Lösen mit AMPL

```
ampl: model fixtransp.mod;
ampl: data fixtransp.dat;
ampl: solve;
MINOS 5.51: ignoring integrality of 18 variables
MINOS 5.51: optimal solution found.
18 iterations, objective 226096
ampl: display X;
X [*,*] (tr)
:    CLEV  GARY  PITT    :=
DET   625     0   575
FRA   275     0   625
FRE   425   625    50
LAF   225   150   625
LAN   600     0     0
STL   450   625   625
WIN     0     0   400
;
```

1.3 Beispiel: Raumzuordnung

Situation:

- n Leute sollen auf n Büros verteilt werden.
- Jede Person i gibt eine Rangfolge der Büros an (r_{ij} ist der Rang von Büro j in der Liste von Person i).
- Aufgabe: Finde eine Zuteilung, so dass die Summe der erhaltenen Ränge minimal ist.

[7]

AMPL-Modell (`assign.mod`)

```
set PEOPLE;
set OFFICES;

param rank {PEOPLE,OFFICES};

var X {PEOPLE,OFFICES} binary;

minimize Total_Cost:
   sum {i in PEOPLE, j in OFFICES}
     rank[i,j] * X[i,j];

subject to assign_p {i in PEOPLE}:
   sum {j in OFFICES} X[i,j] = 1;

subject to assign_o {j in OFFICES}:
   sum {i in PEOPLE} X[i,j] = 1;
```

AMPL-Daten (`myassign.dat`)

```
set PEOPLE := Coullard Daskin Hazen Hopp
              Iravani Linetsky Mehrotra Nelson
              Smilowitz Tamhane White ;

set OFFICES := C118 C138 C140 C246 C250 C251 D237 D239
               D241 M233 M239;

param rank:
          C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239:=
Coullard     6    9    8    7   11   10    4    5    3    2    1
Daskin      11    8    7    6    9   10    1    5    4    2    3
Hazen        9   10   11    1    5    6    2    7    8    3    4
Hopp        11    9    8   10    6    5    1    7    4    2    3
Iravani      3    2    8    9   10   11    1    5    4    6    7
Linetsky    11    9   10    5    3    4    6    7    8    1    2
Mehrotra     6   11   10    9    8    7    1    2    5    4    3
Nelson      11    5    4    6    7    8    1    9   10    2    3
Smilowitz   11    9   10    8    6    5    7    3    4    1    2
Tamhane      5    6    9    8    4    3    7   10   11    2    1
White       11    9    8    4    6    5    3   10    7    2    1;
```

Lösen mit AMPL

```
ampl: model assign.mod;
ampl: data myassign.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 28
26 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl: display X;
X [*,*]
:        C118 C138 C140 C246 C250 C251 D237 D239 D241 M233 M239 :=
Coullard   0    0    0    0    0    0    0    0    1    0    0
Daskin     0    0    0    0    0    0    1    0    0    0    0
Hazen      0    0    0    1    0    0    0    0    0    0    0
Hopp       0    0    0    0    0    1    0    0    0    0    0
Iravani    0    1    0    0    0    0    0    0    0    0    0
Linetsky   0    0    0    0    1    0    0    0    0    0    0
Mehrotra   0    0    0    0    0    0    0    1    0    0    0
Nelson     0    0    1    0    0    0    0    0    0    0    0
Smilowitz  0    0    0    0    0    0    0    0    0    1    0
Tamhane    1    0    0    0    0    0    0    0    0    0    0
White      0    0    0    0    0    0    0    0    0    0    1
;
```