Modellierung in AMPL (Datei `myprod0.mod`)

```
var XB;
var XR;
maximize Profit: 20 * XB + 30 * XR;
subject to Zeit: (1/200) * XB + (1/140) * XR <= 40;
subject to B_Grenzen: 0 <= XB <= 6000;
subject to R_Grenzen: 0 <= XR <= 4000;
```

Lösen in AMPL

```
ampl: model myprod0.mod;
ampl: solve;
MINOS 5.51: optimal solution found.
2 iterations, objective 165714.2857
ampl: display XB;
XB = 2285.71
ampl: display XR;
XR = 4000
```

[4]

AMPL-Modell für beliebige Produkte (myprod.mod)

```
set P;

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

var X {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 (mytransp.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 mytransp.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: option solver gurobi;
ampl: solve;
Gurobi 6.5.0: optimal solution; objective 225396
15 simplex iterations
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
;
```