Cheat-Sheet – Algebraisches Modell zu GAMS-Code

1 Ablaufplanung - Algebraisches Modell Minimierung der Summe der Fertigstellungszeitpunkte[1]

Tabelle 1: Notation des Modells

Indexmengen und Indizes:	
$j \in \mathcal{J} = \{1, ..., J\}$	Jobs
$r \in \mathcal{R} = \{1, ..., R\}$	Ressourcen
$s \in \mathcal{S}_j = \{1, ..., ls_j\}$	Schritte des Jobs j
$t, \tau \in \mathcal{T} = \{1, ..., T\}$	Perioden
Parameter:	
a_{jsr}	benötigte Kapazitätseinheiten der Ressource r für Schritt s des Jobs j
b_{rt}	Kapazität der Ressource r in Periode t
c	Umrechnungsfaktor von Perioden in Stunden
d_{js}	Dauer von Schritt s des Jobs j
ls_j	letzter Schritt von Job j
Entscheidungsvariablen:	
$TF_{js} \in \mathbb{Q} \geq 0$	Fertigstellungszeitpunkt von Schritt s des Jobs j
$X_{jst} \in \{0, 1\}$	binäre Variable mit Wert 1, wenn in Periode t der Schritt s des Jobs j beendet wird, 0 sonst
$Z \in \mathbb{Q}$	Zielfunktionswert

Zielfunktion:

$$\min \quad Z = c \cdot \sum_{j \in \mathcal{J}} TF_{j,ls_j} \tag{1}$$

Nebenbedingungen:

$$\sum_{t=1}^{T} X_{jst} = 1 \qquad \forall j \in \mathcal{J}, s \in \mathcal{S}_j \tag{2}$$

$$TF_{js} \geq TF_{j,s-1} + d_{js} \qquad \forall j \in \mathcal{J}, s \in \{2, ..., ls_j\} \tag{3}$$

$$\sum_{t=1}^{T} t \cdot X_{jst} = TF_{js} \qquad \forall j \in \mathcal{J}, s \in \mathcal{S}_j \tag{4}$$

$$\sum_{j \in \mathcal{J}} \sum_{s \in \mathcal{S}_j} \sum_{\tau=t}^{t+d_{js}-1} a_{jsr} \cdot X_{js\tau} \leq b_{rt} \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{5}$$

[1] vgl. Operations Management Tutorial Aufl.2 S. 223

2 Ablaufplanung - Instanz

Indexmengen und Indizes:
$j \in \mathcal{J} = \{\ j1, j2, j3, j4\ \}$

$r \in \mathcal{R} = \{\ rA, rB, rC\ \}$

$t, \tau \in \mathcal{T} = \{\ t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14, t15, t16, t17, t18, t19, t20\ \}$

$i \in \mathcal{S} = \{\ s1, s2\ \}$

$s \in \mathcal{S}_j = \{\ \mathrm{i} \in \mathcal{S} \mid i \leq ls_j\}$

Parameter:
$b_{r,t} = 1 \quad \forall\, r \in \mathcal{R},\ t \in \mathcal{T}$

$c = 1$

a_{jsr}	rA		rB		rC	
	$s1$	$s2$	$s1$	$s2$	$s1$	$s2$
$j1$	1	0	0	1	0	0
$j2$	1	0	0	0	0	1
$j3$	0	1	1	0	0	0
$j4$	0	0	0	0	1	0

j	ls_j
$j1$	2
$j2$	2
$j3$	2
$j4$	1

d_{js}	$s1$	$s2$
$j1$	3	2
$j2$	1	3
$j3$	3	2
$j4$	4	0

3 Ablaufplanung - GAMS Modell

```
1  Sets
2  j         Jobs
3  r         Ressourcen
4  t         Perioden
5  s         Schritte
6  SJ(s, j)  Schritte des Jobs j
7  ;

9  alias(t, tau);

11 Parameter
12 a(j, s, r) benötigte Kapazitätseinheiten der Ressource r für
       Schritt s des Jobs j
13 b(r, t)    Kapazität der Ressource r in Periode t
14 c          Umrechnungsfaktor von Perioden in Stunden
15 d(j, s)    Dauer von Schritt s des Jobs j
16 ls(j)      letzter Schritt von Job j
17 ;

19 Positive Variables
20 TF(j, s) Fertigstellungszeitpunkt von Schritt s des Jobs j
21 ;

23 Binary Variables
24 X(j, s, t) binäre Variable mit Wert 1 wenn in Periode t der Schritt
       s des Jobs j beendet wird 0 sonst
25 ;

27 Variables
28 Z        Zielfunktionswert
29 ;

31 Equations
32 1_ObjFunc           Minimierung der Gesamtdurchlaufzeit

34 2_Einmal(j, s)       Jeden Arbeitsgang einmal abschliessen
35 3_Schritte(j, s)     Einhaltung der Arbeitsplaene aller Jobs
36 4_Zeitpunkte(j, s)   Kopplung stetiger und binaerer Variablen
37 5_Ressourcen(r, t)   Kapazitätsgrenzen der Ressourcen r zum
       Zeitpunkt t
38 ;

40 1_ObjFunc..
41     Z =e= c * sum((j, s)$(ord(s)=ls(j)), TF(j, s));


44 2_Einmal(j, s)$SJ(s, j)..
45     sum(t, x(j, s, t)) =e= 1;

47 3_Schritte(j, s)$(ord(s)>=2 and ord(s)<=ls(j))..
48     TF(j, s) =g= TF(j, s-1) + d(j, s);

50 4_Zeitpunkte(j, s)$SJ(s, j)..
51     sum(t, ord(t) * x(j, s, t)) =e= TF(j, s);

53 5_Ressourcen(r, t)..
54     sum(j,
55      sum(s$SJ(s, j),
56       sum(tau$((ord(tau)>=ord(t)) and (ord(tau)<=ord(t)+d(j,s)-1)),
57        a(j, s, r) * x(j, s, tau)))) =l= b(r, t);
```

4 Ablaufplanung - Instanz in GAMS

```
sets     j /j1 * j4/
         r /rA,rB,rC/
         t /t1*t20/;
         s /s1*s2/

parameter
ls(j) /j1 2, j2 2, j3 2, j4 1/;

table d(j,s)
                    s1       s2
          j1        3        2
          j2        1        3
          j3        3        2
          j4        4        0       ;

SJ(s, j)=no;

loop(j,
    loop(s,
        if(ord(s) <= ls(j),
                SJ(s, j) = yes;
        );
    );
);


a(j,s,r)=0;
a('j1','s1','rA')=1;
a('j1','s2','rB')=1;

a('j2','s1','rA')=1;
a('j2','s2','rC')=1;

a('j3','s1','rB')=1;
a('j3','s2','rA')=1;

a('j4','s1','rC')=1;


b(r,t)=1;

c = 1;
```

5 Ablaufplanung - Modellerstellung und Lösen in GAMS

```
Model Ablaufplanung /ObjFunc, Einmal, Schritte,Zeitpunkte,
   Ressourcen/;

solve Ablaufplanung minimizing Z using mip;

display Z.l, TF.l, x.l;
```

6 Ablaufplanung - Python Modell

```
#import gurobipy as gp
from gurobipy import *

def Ablaufplanung_model(J, R, S, T, SJ, a, b, c, d, ls):
    model = Model()

    TF = model.addVars(J, S, vtype=GRB.CONTINUOUS, lb=0.0, name='TF')

    X = model.addVars(J, S, T, vtype=GRB.BINARY, name='X')

    #quicksum ist eine Funktion des package "gurobipy"
    obj = c * quicksum(TF[j,ls[j]] for j in J)

    model.setObjective(obj, GRB.MINIMIZE)


    model.addConstrs((quicksum (X[j,s,t] for t in T) == 1 for j in J for s in SJ[j]), 'Einmal')

    for j in J:
        for index_s, s in enumerate(SJ[j]):
            if index_s >= 1:
                model.addConstr((TF[j,s] >= TF[j,S[index_s-1]] + d[j,s]), 'Schritte')

    model.addConstrs((quicksum((t_index+1) * X[j,s,t] for t_index, t in enumerate(T)) == TF[j,s] for j in J for s in SJ[j]), 'Zeitpunkte')

    model.addConstrs((quicksum(a[j,s,r] * X[j,s,tau] for j in J for s in SJ[j] for index_tau, tau in enumerate(T) if index_t <= index_tau <= index_t + d[j,s] - 1) <= b[r,t] for r in R for index_t, t in enumerate(T)), 'Ressourcen')

    return model
```

7 Ablaufplanung - Instanz in Python

```
import CheatSheetmodel
#CheatSheetmodel ist der Name der Datei des Modells
#import gurobipy as gp
from gurobipy import *

Jobs = 4
Ressourcen = 3
Perioden = 20
Schritte = 2

J = [f'j{j}' for j in range(1, Jobs + 1)]

R = ['rA', 'rB', 'rC']

#damit die Schreibweise t1, t2, ... entspricht
T = [f't{t}' for t in range(1, Perioden + 1)]

S = [f's{s}' for s in range(1, Schritte + 1)]

letzte_Schritte = {'j1': 2, 'j2': 2, 'j3': 2, 'j4': 1}

#da die Indexierung in Python mit 0 beginnt, muss der Index des
   letzten Jobs angepasst werden
ls = {}
for job, letzter_Schritt in letzte_Schritte.items():
   ls[job] = S[letzter_Schritt-1]


d = tupledict({('j1', 's1'): 3,
               ('j1', 's2'): 2,
               ('j2', 's1'): 1,
               ('j2', 's2'): 3,
               ('j3', 's1'): 3,
               ('j3', 's2'): 2,
               ('j4', 's1'): 4,
               ('j4', 's2'): 0})

SJ = []
for j in J:
    S_help = []
    for index_i, i in enumerate(S):
        if index_i+1 <= letzte_Schritte[j]:
            S_help.append(i)
    SJ[j] = S_help


a = {(j,s,r): 0 for j in J for s in S for r in R}

a['j1', 's1', 'rA'] = 1
a['j1', 's2', 'rB'] = 1

a['j2', 's1', 'rA'] = 1
a['j2', 's2', 'rC'] = 1

a['j3', 's1', 'rB'] = 1
a['j3', 's2', 'rA'] = 1

a['j4', 's1', 'rC'] = 1
```

```
b = {(r, t): 1 for r in R for t in T}

c = 1
```

8 Ablaufplanung - Modellerstellung und Lösen in Python

```
Ablauf = CheatSheetmodel.Ablaufplanung_model(J, R, S, T, SJ, a, b,
    c, d, ls)

Ablauf.optimize()

def PrintVars(model):
    Vars = {}
    if model.status == GRB.OPTIMAL:
        for variable in model.getVars():
            if variable.x > 0:
                Vars[variable.Varname] = variable.x
        print('Obj: %g' % model.ObjVal)
        print(Vars)
    elif model.status == GRB.INFEASIBLE:
        print('is infeasible')

PrintVars(Ablauf)
```