Energiemodelle und Analysen

Übung 2

Gruppe 1

Technische Universität Wien

Energy Economics Group

Dipl-Ing. Theresia Perger

Wien, Mai 2021

2. Lineare Programmierung

2.1 Herstellung von Produkten aus Recycling-Materialien

Zur Verfügung stehende Kunststoffarten:

Kunststoffarten	Preis (EUR/kg)	Maximal verfügbare Menge (kg)
K_1	5	3000
K_2	6	2000
K_3	4	4000
K_4	5	1000

Produkte und Zusammensetzung:

Produkt	K_1	K_2	K_3	K_4	Preis (EUR/kg)
P_1	50%	30%	20%		8
P_2	40%	30%		30%	7.5
P_3			30%	70%	9

(a) Optimierungsmodell zur Gewinnmaximierung

Es stehen 4 Kunststoffarten zur Produktion von 3 verschiedenen Produkten zur Verfügung. Die Produktion ist jeweils durch die maximal verfügbare Menge begrenzt, siehe (1) – (4) und darf nicht negativ sein (5).

$$0.5 * p_1 + 0.4 * p_2 \leq 3000 \quad (1)$$

$$0.3 * p_1 + 0.3 * p_2 \leq 2000 \quad (2)$$

$$0.2 * p_1 + 0.3 * p_3 \leq 4000 \quad (3)$$

$$0.3 * p_2 + 0.7 * p_3 \leq 1000 \quad (4)$$

$$p_1, p_2, p_3 \geq 0 \quad (5)$$

(1) – (5) sind die jeweiligen Nebenbedingungen.

Der jeweilige Gewinn errechnet sich folgendermaßen:

$$Gp_1 = R - C = Rp_1 - Ck_1 * 0.5 - Ck_2 * 0.3 - Ck_3 * 0.2 = 3.4 \quad (6)$$

$$Gp_2 = Rp_2 - Ck_1 * 0.4 - Ck_2 * 0.3 - Ck_4 * 0.3 = 2.6 \quad (7)$$

$$Gp_3 = Rp_3 - Ck_3 * 0.3 - Ck_4 * 0.7 = 4.3 \quad (8)$$

Die folgende Gewinnfunktion wird all Zielfunktion maximiert:

$$G = 3.4 * p_1 + 2.6 * p_2 + 4.3 * p_3 \quad (9)$$

(b) Lösung des Optimierungsproblems mit Hilfe des Simplex-Algorithmus

Die Nebenbedingungen werden jeweils um eine Schlupfvariable erweitert und in eine Gleichung übertragen damit es mittels Simplexverfahren gelöst werden kann.

$$0.5 * p_1 + 0.4 * p_2 + u_1 = 3000 \quad (10)$$

$$0.3 * p_1 + 0.3 * p_2 + u_2 = 2000 \quad (11)$$

$$0.2 * p_1 + 0.3 * p_3 + u_3 = 4000 \quad (12)$$

$$0.3 * p_2 + 0.7 * p_3 + u_4 = 1000 \quad (13)$$

Diese werden dann im nächsten Schritt ins Simplex Tableau eingetragen:

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b
u_1	0.5	0.4	0	1	0	0	0	3000
u_2	0.3	0.3	0	0	1	0	0	2000
u_3	0.2	0	0.3	0	0	1	0	4000
u_4	0	0.3	0.7	0	0	0	1	1000
	3.4	2.6	4.3	0	0	0	0	

Danach wird in der letzten Zeile der größte Wert gesucht und die Pivotspalte definiert. Die Werte von b werden im nächsten Schritt durch die Werte der Pivotspalte dividiert und der kleinste Wert als Pivotzeile definiert.

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	b/Piv.
u_1	0.5	0.4	0	1	0	0	0	3000	∞
u_2	0.3	0.3	0	0	1	0	0	2000	∞
u_3	0.2	0	0.3	0	0	1	0	4000	13333.33
u_4	0	0.3	0.7	0	0	0	1	1000	1428.57
	3.4	2.6	4.3	0	0	0	0	0	

Als nächstes wird die gesamte Pivotzeile durch das Piovotelement (Kreuzung aus Zeile und Spalte) dividiert.

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	b/Piv.
u_1	0.5	0.4	0	1	0	0	0	3000	
u_2	0.3	0.3	0	0	1	0	0	2000	
u_3	0.2	0	0.3	0	0	1	0	4000	
p_3	0	0.43	1	0	0	0	1.43	1428.57	
	3.4	2.6	4.3	0	0	0	0	0	

Alle anderen Elemente der Pivoltspalte werden auf 0 gebracht

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	
u_1	0.5	0.4	0	1	0	0	0	3000	
u_2	0.3	0.3	0	0	1	0	0	2000	
u_3	0.2	-0,13	0	0	0	1	-0,43	3571.43	Z3-Z4*0.3
p_3	0	0.43	1	0	0	0	1.43	1428.57	

	3.4	0.75	0	0	0	0	-6.15	-6142.85	Z5-Z4*4,3

Das genannte Vorgehen wird so lange wiederholt bis alle Werte in der letzten Zeile negativ sind.

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	b/Piv.
p_1	0.5	0.4	0	1	0	0	0	3000	6000
u_2	0.3	0.3	0	0	1	0	0	2000	6666.67
u_3	0.2	-0,13	0	0	0	1	-0,43	3571.43	17857.15
p_3	0	0.43	1	0	0	0	1.43	1428.57	∞
	3.4	0.75	0	0	0	0	-6.15	-6142.85	

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	
p_1	1	0.8	0	2	0	0	0	6000	
u_2	0.3	0.3	0	0	1	0	0	2000	
u_3	0.2	-0,13	0	0	0	1	-0,43	3571.43	
p_3	0	0.43	1	0	0	0	1.43	1428.57	
	3.4	0.75	0	0	0	0	-6.15	-6142.85	

	p_1	p_2	p_3	u_1	u_2	u_3	u_4	b	
p_1	1	0.8	0	2	0	0	0	6000	
u_2	0	0.06	0	-0.6	1	0	0	200	Z2-Z1*0,3
u_3	0	-0.29	0	-0.4	0	1	-0.43	2371.43	Z3-Z1*0,2
p_3	0	0.43	1	0	0	0	1.43	1428.57	
	0	-1.96	0	-6.8	0	0	-6.15	-26542.9	Z5-Z1*3,4

Da nun alle Werte der Zielfunktion negativ sind kann man folgende Ergebnisse ablesen:

P_1 = 6000 kg

P_2 = 0 kg

P_3 = 1428.57 kg

Mit einem maximalen Gesamtgewinn von 26543 EUR.

2.2 A)

Mathematical Formula

Variables

Constraints ¶

$x_1 + x_2 + x_3 >= 125$

$0 <= x_1 <= 30$

$0 <= x_2 <= 60$

$0 <= x_3 <= 40$

Case (i) Objective Function:

$Cost = SUM_i (X_i * (VarCost_i + FuelCost_i / \eta_i))$

Case (ii) Objective Function:

$Emm = SUM_i (X_i * EmmFactor_i / \eta_i)$

2.2 b)

	x1	x2	x3	x4	x5	x6	x7	a1	b	
U1	1	1	1	-1	0	0	0	1	125	x1,x2,x3,x4=0
U2	1	0	0	0	1	0	0	0	30	Basislösung
U3	0	1	0	0	0	1	0	0	60	a1=125
U4	0	0	1	0	0	0	1	0	40	x5=30
Z	0	0	0	0	0	0	0	-1		x6=60
										x7=40

Punkt 3: Addition aller Zeilen die eine Hilfsvariable enthalten

	x1	x2	x3	x4	x5	x6	x7	a1	b	b/Piv
U1	1	1	1	-1	0	0	0	1	125	125
U2	1	0	0	0	1	0	0	0	30	30
U3	0	1	0	0	0	1	0	0	60	#DIV/0!
U4	0	0	1	0	0	0	1	0	40	#DIV/0!
Z	1	1	1	-1	0	0	0	0	125	

Anwenden des Simplex Verfahrens

x1	x2	x3	x4	x5	x6	x7	a1	b	b/Piv
0	1	1	-1	-1	0	0	1	95	95
1	0	0	0	1	0	0	0	30	#DIV/0!
0	1	0	0	0	1	0	0	60	60
0	0	1	0	0	0	1	0	40	#DIV/0!
0	1	1	-1	-1	0	0	0	95	

x1	x2	x3	x4	x5	x6	x7	a1	b	b/Piv
0	0	1	-1	-1	-1	0	1	35	35
1	0	0	0	1	0	0	0	30	#DIV/0!
0	1	0	0	0	1	0	0	60	#DIV/0!
0	0	1	0	0	0	1	0	40	40
0	0	1	-1	-1	-1	0	0	35	

x1	x2	x3	x4	x5	x6	x7	a1	b	b/Piv
0	0	1	-1	-1	-1	0	1	35	
1	0	0	0	1	0	0	0	30	
0	1	0	0	0	1	0	0	60	
0	0	0	1	1	1	1	-1	5	
0	0	0	0	0	0	0	-1	0	

z* ist minimal mit 0 Einsetzen der echten Zielfunktion:

	x1	x2	x3	x4	x5	x6	x7	b	b/Piv
U1	0	0	1	-1	-1	-1	0	35	
U2	1	0	0	0	1	0	0	30	
U3	0	1	0	0	0	1	0	60	
U4	0	0	0	1	1	1	1	5	
Z	-24.72	-66.92	-77.9	0	0	0	0	0	

Durch Addition eines Vielfachen von geeigneten Zeilen zur Zielfunktionszeile bringen wir alle Einträge der Basisvariablen in der Zielfunktionszeile auf Null.

	x1	x2	x3	x4	x5	x6	x7	b
U1	0	0	1	-1	-1	-1	0	35
U2	1	0	0	0	1	0	0	30
U3	0	1	0	0	0	1	0	60
U4	0	0	0	1	1	1	1	5
Z	0	-66.92	-77.9	0	24.72	0	0	741.6

	x1	x2	x3	x4	x5	x6	x7	b
U1	0	0	1	-1	-1	-1	0	35
U2	1	0	0	0	1	0	0	30
U3	0	1	0	0	0	1	0	60
U4	0	0	0	1	1	1	1	5
Z	0	0	-77.9	0	24.72	66.92	0	4756.8

	x1	x2	x3	x4	x5	x6	x7	b
U1	0	0	1	-1	-1	-1	0	35
U2	1	0	0	0	1	0	0	30
U3	0	1	0	0	0	1	0	60
U4	0	0	0	1	1	1	1	5
Z	0	0	0	-77.9	-53.18	-10.98	0	7483.3

Aufgabe 2.3 - Schadstoffreduzierung eines Stahlwerks

In [6]:
```
import pyomo.environ as pyo
```

a)

Die Zielfunktion für das Optimierungsproblem ergibt sich zu:

$$min \sum A_{j,i} * x_i * ci$$

wobei A die Matrix für die möglich Emmisionsreduktion pro Technologie und Hochofen dar stellt, x der prozentuelle Einsatz der Technologie ist, um c die jeweiligen Kosten für die Anwendung einer Technologie.

A in 1000 Tonnen:

Schadstoff	Schornsteine 1	Schornsteine 2	Filter 1	Filter 2	Brennstoffe 1	Brennstoffe 2
Staub Ruß	12	9	25	20	17	13
Schwefeloxid	35	42	18	31	56	49
Kohlenwasserstoffe	37	53	28	24	29	30

c in Mio. EUR:

	Schornsteine 1	Schornsteine 2	Filter 1	Filter 2	Brennstoffe 1	Brennstoffe 2
Kosten	8	10	7	6	11	9

x in %:

	Schornsteine 1	Schornsteine 2	Filter 1	Filter 2	Brennstoffe 1	Brennstoffe 2
Prozent	?	?	?	?	?	?

Die Nebenbedingung ist:

$$b <= A_{j,i} * x_i$$

mit b als Paramter der Schadstoffe die mindestens eingespart werden müssen (in 1000 Tonnen):

Staub Russ	Schwefeloxid	Kohlenwasserstoffe
60	150	125

Im folgenden wird der Code für das Model bereit gestellt und die Lösung für a), b) und c) dargestellt.

In [2]:
```
# Aufbereiten der Modelleingangsparameter:
A = [[12,9,25,20,17,13],
     [35,42,18,31,56,49],
     [37,53,28,24,29,20]]
b = [60,150,125]
c = [8, 10,7,6,11,9]

Schadstoffe = ["staub_russ", "schwefel", "CH"]

Masnahmen = ["Schornstein_1", "Schornstein_2",
             "Filter_1", "Filter_2",
             "Brennstoff_1", "Brennstoff_2"]


dictionary = {}
reduktion = {}
cost = {}
for reihe, stoff in enumerate(Schadstoffe):
    reduktion[stoff] = b[reihe]
    for spalte, mas in enumerate(Masnahmen):
        dictionary[(stoff, mas)] = A[reihe][spalte]

for i, mas in enumerate(Masnahmen):
    cost[mas] = c[i]
```

In [3]:
```
# erstellen des Models + Lösung b):

model = pyo.AbstractModel()

model.schadstoffe = pyo.Set(initialize=Schadstoffe)
model.masnahmen = pyo.Set(initialize=Masnahmen)


model.A = pyo.Param(model.schadstoffe, model.masnahmen, initialize=dict
model.b = pyo.Param(model.schadstoffe, initialize=reduktion)
model.c = pyo.Param(model.masnahmen, initialize=cost)

model.x = pyo.Var(model.masnahmen, bounds=(0,1), within=pyo.NonNegative


def zielfunktion(model):
    return sum(model.A[j,i] * model.x[i] * model.c[i] for j in model.sc
model.cost = pyo.Objective(rule=zielfunktion, sense=pyo.minimize)


def schadstoff_rule(model, j):
        value = sum(model.A[j,i] * model.x[i] for i in model.masnahmen)
        return model.b[j] <= value
model.Nebenbedingung = pyo.Constraint(model.schadstoffe, rule=schadstof


instance = model.create_instance()
opt = pyo.SolverFactory("gurobi")
results = opt.solve(instance)

instance.display()
```

```
Model unknown

  Variables:
    x : Size=6, Index=masnahmen
        Key           : Lower : Value                : Upper : Fixed
: Stale : Domain
         Brennstoff_1 :     0 : 0.04757281553398047 :     1 : False
: False : NonNegativeReals
         Brennstoff_2 :     0 :                1.0 :     1 : False
: False : NonNegativeReals
             Filter_1 :     0 : 0.34347940173182906 :     1 : False
: False : NonNegativeReals
             Filter_2 :     0 :                1.0 :     1 : False
: False : NonNegativeReals
        Schornstein_1 :     0 :                1.0 :     1 : False
: False : NonNegativeReals
        Schornstein_2 :     0 :  0.6226974547362896 :     1 : False
: False : NonNegativeReals

  Objectives:
    cost : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True : 2731.691314615586

  Constraints:
    Nebenbedingung : Size=3
        Key         : Lower : Body  : Upper
                CH : 125.0 : 125.0 :  None
          schwefel : 150.0 : 150.0 :  None
        staub_russ :  60.0 :  60.0 :  None
```

In [4]:

```
# erstellen des Models + Lösung c):

model = pyo.AbstractModel()

model.schadstoffe = pyo.Set(initialize=Schadstoffe)
model.masnahmen = pyo.Set(initialize=Masnahmen)


model.A = pyo.Param(model.schadstoffe, model.masnahmen, initialize=dict
model.b = pyo.Param(model.schadstoffe, initialize=reduktion)
model.c = pyo.Param(model.masnahmen, initialize=cost)

model.x = pyo.Var(model.masnahmen, bounds=(0,1), within=pyo.Binary)


def zielfunktion(model):
    return sum(model.A[j,i] * model.x[i] * model.c[i] for j in model.sc
model.cost = pyo.Objective(rule=zielfunktion, sense=pyo.minimize)


def schadstoff_rule(model, j):
        value = sum(model.A[j,i] * model.x[i] for i in model.masnahmen)
        return model.b[j] <= value
model.Nebenbedingung = pyo.Constraint(model.schadstoffe, rule=schadstof


instance_2 = model.create_instance()
opt = pyo.SolverFactory("gurobi")
results = opt.solve(instance_2)
```

```
instance_2.display()
Model unknown

  Variables:
    x : Size=6, Index=masnahmen
        Key           : Lower : Value : Upper : Fixed : Stale : Doma
in
         Brennstoff_1 :     0 :   1.0 :     1 : False : False : Bina
ry
         Brennstoff_2 :     0 :  -0.0 :     1 : False : False : Bina
ry
             Filter_1 :     0 :   1.0 :     1 : False : False : Bina
ry
             Filter_2 :     0 :  -0.0 :     1 : False : False : Bina
ry
        Schornstein_1 :     0 :   1.0 :     1 : False : False : Bina
ry
        Schornstein_2 :     0 :   1.0 :     1 : False : False : Bina
ry

  Objectives:
    cost : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True : 3331.0

  Constraints:
    Nebenbedingung : Size=3
        Key        : Lower : Body  : Upper
                CH : 125.0 : 147.0 :  None
          schwefel : 150.0 : 151.0 :  None
        staub_russ :  60.0 :  63.0 :  None
```

Damit ergeben sich für Aufgabe 2.3.3 b) und c):

b) x_b in %:

	Schornsteine 1	Schornsteine 2	Filter 1	Filter 2	Brennstoffe 1	Brennstoffe 2
Prozent	100	62.27	34.35	100	4.75	100

c) x_c in %:

	Schornsteine 1	Schornsteine 2	Filter 1	Filter 2	Brennstoffe 1	Brennstoffe 2
Prozent	100	100	100	0	100	0

d) Wenn nur ein Typ für jede Maßnahme eingesetzt werden kann ist das Problem unlösbar, da die Nebenbedingung (erforderlichen Reduktionen) nicht erreicht werden kann: Das Programm gibt die Fehlermeldung: "*message from solver: Model was proven to be infeasible*" aus.

In [5]:
```
# erstellen des Models + Lösung d):

model = pyo.AbstractModel()
```

```
model.schadstoffe = pyo.Set(initialize=Schadstoffe)
model.masnahmen = pyo.Set(initialize=Masnahmen)


model.A = pyo.Param(model.schadstoffe, model.masnahmen, initialize=dict
model.b = pyo.Param(model.schadstoffe, initialize=reduktion)
model.c = pyo.Param(model.masnahmen, initialize=cost)

model.x = pyo.Var(model.masnahmen, bounds=(0,1), within=pyo.Binary)


def zielfunktion(model):
    return sum(model.A[j,i] * model.x[i] * model.c[i] for j in model.sc
model.cost = pyo.Objective(rule=zielfunktion, sense=pyo.minimize)


def schadstoff_rule(model, j):
        value = sum(model.A[j,i] * model.x[i] for i in model.masnahmen)
        return model.b[j] <= value
model.Nebenbedingung = pyo.Constraint(model.schadstoffe, rule=schadstof


def extra_nebenbedingung_schornstein(model):
    return model.x["Schornstein_1"] + model.x["Schornstein_2"] <= 1
model.extra_NB_schornstein = pyo.Constraint(rule=extra_nebenbedingung_s


def extra_nebenbedingung_filter(model):
    return model.x["Filter_1"] + model.x["Filter_2"] <= 1
model.extra_NB_filter = pyo.Constraint(rule=extra_nebenbedingung_filter


def extra_nebenbedingung_brennstoff(model):
    return model.x["Brennstoff_1"] + model.x["Brennstoff_2"] <= 1
model.extra_NB_brennstoff = pyo.Constraint(rule=extra_nebenbedingung_br


instance_2 = model.create_instance()
opt = pyo.SolverFactory("gurobi")
results = opt.solve(instance_2)
```

```
instance_2.display()
WARNING: Loading a SolverResults object with a warning status into
    model.name="unknown";
      - termination condition: infeasible
      - message from solver: Model was proven to be infeasible.
Model unknown

  Variables:
    x : Size=6, Index=masnahmen
        Key           : Lower : Value : Upper : Fixed : Stale : Doma
in
         Brennstoff_1 :     0 :  None :     1 : False :  True : Bina
ry
         Brennstoff_2 :     0 :  None :     1 : False :  True : Bina
ry
             Filter_1 :     0 :  None :     1 : False :  True : Bina
ry
             Filter_2 :     0 :  None :     1 : False :  True : Bina
ry
        Schornstein_1 :     0 :  None :     1 : False :  True : Bina
```

```
ry
        Schornstein_2 :     0 :  None :     1 : False :  True : Bina
ry

  Objectives:
    cost : Size=1, Index=None, Active=True
ERROR: evaluating object as numeric value: x[Schornstein_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Schornstein_1]
ERROR: evaluating object as numeric value: cost
        (object: <class 'pyomo.core.base.objective.SimpleObjective'
>)
    No value for uninitialized NumericValue object x[Schornstein_1]
        Key : Active : Value
        None :   None :  None

  Constraints:
    Nebenbedingung : Size=3
ERROR: evaluating object as numeric value: x[Schornstein_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Schornstein_1]
ERROR: evaluating object as numeric value: x[Schornstein_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Schornstein_1]
ERROR: evaluating object as numeric value: x[Schornstein_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Schornstein_1]
        Key : Lower : Body : Upper
         CH :  None : None :  None
        schwefel :  None : None :  None
        staub_russ :  None : None :  None
    extra_NB_schornstein : Size=1
ERROR: evaluating object as numeric value: x[Schornstein_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Schornstein_1]
        Key : Lower : Body : Upper
        None :  None : None :  None
    extra_NB_filter : Size=1
ERROR: evaluating object as numeric value: x[Filter_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Filter_1]
        Key : Lower : Body : Upper
        None :  None : None :  None
    extra_NB_brennstoff : Size=1
ERROR: evaluating object as numeric value: x[Brennstoff_1]
        (object: <class 'pyomo.core.base.var._GeneralVarData'>)
    No value for uninitialized NumericValue object x[Brennstoff_1]
        Key : Lower : Body : Upper
        None :  None : None :  None
```