Python Kurs 2019/2020

10: Lineare Programmierung in Python

Bernhard Mallinger

`b.mallinger [at] gmx.at`

https://totycro.github.io/python-kurs

Pulp

- `pulp` bietet ein einfaches Interface für LP
- Beschränkt auf lineare Constraints, dadurch einfach
- Volle Ausdruckskraft von Python zum Definieren der Bedingungen (z.B. Daten mittels Pandas dafür auslesen)
- `pulp` mittels Thonny installieren
- Unterstützt verschiedene Solver

Pulp

Problemdefinition

```
from pulp import *
# Problem definieren
prob = LpProblem("problem_name", LpMaximize)  # oder: LpMinimize
```

Variablen definieren

```
x = LpVariable("x", lowBound=0, upBound=1, cat=LpContinuous)
y = LpVariable("y", lowBound=-10, upBound=10, cat=LpInteger)
```

- `bounds` per default negativ bzw. positiv unendlich
- `cat` per default continuous, auch `LpBinary` und `LpInteger` möglich

Pulp

Nebenbedingungen definieren

```
# "zu Problem hinzufügen"
prob += x - 3 * y >= 6
prob += x + 2 * y <= 1
```

Zielfunktion definieren

```
# "zu Problem hinzufügen"
prob += 2 * x + y
```

Unterschied zwischen Zielfunktion und Nebenbedingung: Vergleich

Pulp

Lösung berechnen

```
status = prob.solve()
print(LpStatus[status])

for v in prob.variables():
    print(v.name, "=", v.varValue)

# ähnlich:
print(x.varValue)
```

Pulp: Variablencollections

```
bindertypen = list(range(1, 4))
umgebungsbedingungen = list(range(1, 4))

hobelmass = LpVariable.dict("hobelmass", indexs=bindertypen)
# {1: hobelmass_1, 2: hobelmass_2, 3: hobelmass_3}

hobelmass = LpVariable.dict(
    "hobelmass",
     indexs=(bindertypen, umgebungsbedingungen),
 )
# {(1, 1): hobelmass_1_1, (1, 2): hobelmass_1_2, (1, 3): hobelmass_1_3,

hobelmass = LpVariable.dicts(
    "hobelmass",
     indexs=(bindertypen, umgebungsbedingungen),
 )
# {1: {1: hobelmass_1_1, 2: hobelmass_1_2, 3: hobelmass_1_3}, 2: {1: ho
```

Pulp: Beispiele

```
# Problemdefinition als normaler Python-Code
for bt in bindertypen:
    for ub in umgebungsbedingungen:
        prob += hobelmass[bt][ub] > 2
```

```
choice_costs = {'a': 10, 'b': 7, 'c': 3}

# Beispiel: Auswahl auf genau 1 setzen mittels lpSum
choice_vars = LpVariable.dict("choices", choice_costs, cat=LpBinary)
prob += lpSum(choice_vars) == 1
```

```
# Beispiel: Zielfunktion
prob += lpSum(
    [
        choice_cost * choice_vars[choice]
        for choice, choice_cost in choice_costs.items()
    ]
)
```

Pulp: Debugging

```
>>> print(prob)
problem_name:
MAXIMIZE
2*x + 1*y + 0
SUBJECT TO
_C1: x - 3 y >= 6

_C2: x + 2 y <= 1
[...]
```

Tipp: Namen für Nebenbedingungen vergeben

```
prob += x + y >= 3, "Special condition"
```

```
>>> print(prob)
[...]
Special_condition: x + y >= 3
[...]
```