LpVariable dictionary function

SUPPLY CHAIN ANALYTICS IN PYTHON

Aaren Stubberfield
Supply Chain Analytics Mgr.

Moving from simple to complex

Complex Bakery Example

```
# Define Decision Variables
A = LpVariable('A', lowBound=0, cat='Integer')
B = LpVariable('B', lowBound=0, cat='Integer')
C = LpVariable('C', lowBound=0, cat='Integer')
D = LpVariable('D', lowBound=0, cat='Integer')
E = LpVariable('E', lowBound=0, cat='Integer')
F = LpVariable('F', lowBound=0, cat='Integer')
```

```
# Define Objective Function
var_dict = {"A":A, "B":B, "C":C, "D":D, "E":E, "F":F}

# Define Objective Function
model += lpSum([profit_by_cake[type] * var_dict[type] for type in cake_types])
```

Using LpVariable.dicts()

```
LpVariable(name, indexs, lowBound=None, upBound=None, cat='Continuous')
```

- `name` = The prefix to the name of each LP variable created
- `indexs` = A list of strings of the keys to the dictionary of LP variables
- `lowBound` = Lower bound
- `upBound` = Upper bound
- `cat` = The type of variable this is
  - Integer
  - Binary
  - Continuous *(default)*

LpVariable.dicts() with list comprehension

- `LpVariable.dicts()` often used with Python's list comprehension

Transportation Optimization

```
# Define Decision Variables
customers = ['East','South','Midwest','West']
warehouse = ['New York','Atlanta']
transport = LpVariable.dicts("route", [(w,c) for w in warehouse for c in customers],
                              lowBound=0, cat='Integer')
# Define Objective
model += lpSum([cost[(w,c)]*transport[(w,c)] for w in warehouse for c in customers])
```

Summary

- Creating many LP variables for complex problems
- `LpVariable.dicts()`
- Used with list comprehension

Now you try it out

SUPPLY CHAIN ANALYTICS IN PYTHON

Example of a scheduling problem

SUPPLY CHAIN ANALYTICS IN PYTHON

Aaren Stubberfield
Supply Chain Analytics Mgr.

datacamp

Expected Demand

Day of Week	Drivers Needed
0 = Monday	11
1= Tuesday	14
2 = Wednesday	23
3 = Thursday	21
4 = Friday	20
5 = Saturday	15
6 = Sunday	8

Question:

- How many drivers, in total, do we need to hire?

Constraint:

- Each driver works for 5 consecutive days, followed by 2 days off, repeated weekly

Step	Definition
Decision Var	X_i = the number of drivers working on day _i_
Objective	*minimize* z = $X_0 + X_1 + X_2 + X_3 + X_4 + X_5 + X_6$
Subject to	$X_0 \geq 11$
	$X_1 \geq 14$
	$X_2 \geq 23$
	$X_3 \geq 21$
	$X_4 \geq 20$
	$X_i \geq 0$ (i = 0, ..., 6)

Step	Definition
Decision Var	X_i = the number of drivers working on day _i_
Objective	minimize $z = X_0 + X_1 + X_2 + X_3 + X_4 + X_5 + X_6$
Subject to	$X_0 + X_3 + X_4 + X_5 + X_6 \geq 11$
	$X_0 + X_1 + X_4 + X_5 + X_6 \geq 14$
	$X_0 + X_1 + X_2 + X_3 + X_6 \geq 23$
	$X_0 + X_1 + X_2 + X_3 + X_4 \geq 21$
	$X_1 + X_2 + X_3 + X_4 + X_5 \geq 15$
	X_i ? 0 (i = 0, ..., 6)

Coding example

```python
# Initialize Class
model = LpProblem("Minimize Staffing",
                  LpMinimize)
days = list(range(7))

# Define Decision Variables
x = LpVariable.dicts('staff_', days,
                     lowBound=0, cat='Integer')

# Define Objective
model += lpSum([x[i] for i in days])
```

```python
# Define Constraints
model += x[0] + x[3] + x[4] + x[5] + x[6] >= 11
model += x[0] + x[1] + x[4] + x[5] + x[6] >= 14
model += x[0] + x[1] + x[2] + x[5] + x[6] >= 23
model += x[0] + x[1] + x[2] + x[3] + x[6] >= 21
model += x[0] + x[1] + x[2] + x[3] + x[4] >= 20
model += x[1] + x[2] + x[3] + x[4] + x[5] >= 15
model += x[2] + x[3] + x[4] + x[5] + x[6] >= 8

# Solve Model
model.solve()
```

Summary

- Our initial variables did not work
- Decision variables to incorporate some of the constraints

Practice time!

SUPPLY CHAIN ANALYTICS IN PYTHON

Capacitated plant location - case study P1

SUPPLY CHAIN ANALYTICS IN PYTHON

Aaren Stubberfield
Supply Chain Analytics Mgr.

Context

Multiple options to meet regional product demand

Option	Pro	Con
Small manufacturing facilities within region	Low transportation costs, few to no tariffs or duties	Overall network may have excess capacity, cannot take advantage economies of scale
A few large manufacturing plants and ship product to region	Economies of scale	Higher transportation, higher tariffs and duties

Capacitated plant location model

- Capacitated Plant Location Model[1]
- The goal is to optimize global Supply Chain network
  - Meet regional demand at the lowest cost
  - Determine regional production of a product

[1] Chopra, Sunil, and Peter Meindl. _Supply Chain Management: Strategy, Planning, and Operations._ Pearson Prentice-Hall, 2007.

Capacitated plant location model

Modeling

- Production at regional facilities
  - Two plant sizes (low / high)
- Exporting production to other regions
- Production facilities open / close

Decision variables

What we can control:

- x_{ij} = quantity produced at location **_i_** and shipped to **_j_**
- y_{is} = 1 if the plant at location **_i_** of capacity **_s_** is open, 0 if closed
  - s = ***low*** or ***high*** capacity plant

Objective function

Minimize $z = \sum_{i=1}^{n}(f_{is}y_{is}) + \sum_{i=1}^{n}\sum_{i=1}^{m}(c_{ij}x_{ij})$

- c_{ij} = cost of producing and shipping from plant **_i_** to region **_j_**
- f_{is} = fixed cost of keeping plant **_i_** of capacity **_s_** open
- n = number of production facilities
- m = number of markets or regional demand points

```python
from pulp import *

# Initialize Class
model = LpProblem("Capacitated Plant Location Model", LpMinimize)

# Define Decision Variables
loc = ['A', 'B', 'C', 'D', 'E']
size = ['Low_Cap','High_Cap']
x = LpVariable.dicts("production",[(i,j) for i in loc for j in loc],
                     lowBound=0, upBound=None, cat='Continous')
y = LpVariable.dicts("plant",[(i,s) for s in size for i in loc], cat='Binary')
# Define objective function
model += (lpSum([fix_cost.loc[i,s]*y[(i,s)] for s in size for i in loc])
          + lpSum([var_cost.loc[i,j]*x[(i,j)] for i in loc for j in loc]))
```

Summary

Capacitated Plant Location Model:

- Finds a balance between the number of production facilities
- Model decision variables:
  - Quantity of production in a region and exported
  - High or low capacity facilities open or closed
- Reviewed objective function
  - Sums variable and fixed production costs
- Reviewed code example

Review time

SUPPLY CHAIN ANALYTICS IN PYTHON

Logical constraints

SUPPLY CHAIN ANALYTICS IN PYTHON

Aaren Stubberfield
Supply Chain Analytics Mgr.

datacamp

Example problem

Maximum Weight 20,000 lbs

Product	Weight (lbs)	Profitability ($US)
A	12,800	77,878
B	10,900	82,713
C	11,400	82,728
D	2,100	68,423
E	11,300	84,119
F	2,300	77,765

- Select most profitable product to ship without exceeding weight limit
- Decision Variables:
  - X_i = 1 if product _i_ is selected else 0
- Objective:
  - Maximize z = $\sum$ Profitability$_i$X$_i$
- Constraint:
  - $\sum$ Weight$_i$X$_i$ < 20,0000

```python
prod = ['A', 'B', 'C', 'D', 'E', 'F']
weight = {'A':12800, 'B':10900, 'C':11400, 'D':2100, 'E':11300, 'F':2300}
prof = {'A':77878, 'B':82713, 'C':82728, 'D':68423, 'E':84119, 'F':77765}
# Initialize Class
model = LpProblem("Loading Truck Problem", LpMaximize)

# Define Decision Variables
x = LpVariable.dicts('ship_', prod, cat='Binary')
# Define Objective
model += lpSum([prof[i]*x[i] for i in prod])

# Define Constraint
model += lpSum([weight[i]*x[i] for i in prod]) <= 20000
# Solve Model
model.solve()
for i in prod: print("{} status {}".format(i, x[i].varValue))
```

Example result

Maximum Weight 20,000 lbs

Product	Ship or Not
A	No
B	No
C	No
D	Yes
E	Yes
F	Yes

Result

- Profitability: $230,307
- Weight of Products: 15,700 lbs

Logical constraint example 1

Either product E is selected or product D is selected, but not both.

- X_E = 1 if product _i_ is selected else 0
- X_D = 1 if product _i_ is selected else 0
- Constraint
  - $X_E + X_D \leq 1$

Code example - logical constraint example 1

```
model += x['E'] + x['D'] <= 1
```

```
prod = ['A', 'B', 'C', 'D', 'E', 'F']
weight = {'A':12800, 'B':10900, 'C':11400,
          'D':2100, 'E':11300, 'F':2300}
prof = {'A':77878, 'B':82713, 'C':82728,
        'D':68423, 'E':84119, 'F':77765}

# Initialize Class
model = LpProblem("Loading Truck Problem",
                  LpMaximize)

# Define Decision Variables
x = LpVariable.dicts('ship_', prod,
                     cat='Binary')
```

```
# Define Objective
model += lpSum([prof[i]*x[i] for i in prod])

# Define Constraint
model +=
  lpSum([weight[i]*x[i] for i in prod]) <= 20000
model += x['E'] + x['D'] <= 1

# Solve Model
model.solve()
for i in prod:
  print("{} status {}".format(i, x[i].varValue))
```

Logical constraint 1 example result

Maximum Weight 20,000 lbs

Product	Ship or Not
A	No
B	No
C	Yes
D	Yes
E	No
F	Yes

Result

- Profitability: $228,916
- Weight of Products: 15,800 lbs

Logical constraint example 2

If product D is selected then product B must also be selected.

- X_D = 1 if product _i_ is selected else 0
- X_B = 1 if product _i_ is selected else 0
- Constraint
  - $X_D \leq X_B$

Code example - logical constraint example 2

```
model += x['D'] <= x['B']
prod = ['A', 'B', 'C', 'D', 'E', 'F']
weight = {'A':12800, 'B':10900, 'C':11400,
          'D':2100, 'E':11300, 'F':2300}
prof = {'A':77878, 'B':82713, 'C':82728,
        'D':68423, 'E':84119, 'F':77765}

# Initialize Class
model = LpProblem("Loading Truck Problem",
                  LpMaximize)

# Define Decision Variables
x = LpVariable.dicts('ship_', prod,
                     cat='Binary')
```

```
# Define Objective
model += lpSum([prof[i]*x[i] for i in prod])

# Define Constraint
model +=
 lpSum([weight[i]*x[i] for i in prod]) <= 20000
model += x['D'] <= x['B']

# Solve Model
model.solve()
for i in prod:
  print("{} status {}".format(i, x[i].varValue))
```

Logical constraint 2 example result

Maximum Weight 20,000 lbs

Product	Ship or Not
A	No
B	Yes
C	No
D	Yes
E	No
F	Yes

Result

- Profitability: $228,901
- Weight of Products: 15,300 lbs

Other logical constraints

Logical Constraint	Constraint
If item _i_ is selected, then item _j_ is also selected.	$x_i - x_j \le 0$
Either item _i_ is selected or item _j_ is selected, but not both.	$x_i + x_j = 1$
If item _i_ is selected, then item _j_ is not selected.	$x_i - x_j \le 1$
If item _i_ is not selected, then item _j_ is not selected.	$-x_i + x_j \le 0$
At most one of items _i_, _j_, and _k_ are selected.	$x_i + x_j + x_k \le 1$

[1] James Orlin, and Ebrahim Nasrabadi. 15.053 Optimization Methods in Management Science. Spring 2013. Massachusetts Institute of Technology: MIT OpenCourseWare. License: Creative Commons BY-NC-SA.

Summary

- Reviewed examples of logical constraints
- Listed a table of other logical constraints

Your turn!

SUPPLY CHAIN ANALYTICS IN PYTHON