Week 11: Abstract Formulation

Session 21: Expressing Patterns Precisely using Mathematical Notation

Understanding Summation Notation

In mathematics, sums can be denoted using $\sum$.

Basic Example

```
[8]: from gurobipy import Model, GRB
     mod=Model()
     x=mod.addVars(range(1,11),name='x')
     mod.update()
     B_L=[1,4,5,9]
     sum(x[b] for b in B_L)
```

```
<gurobi.LinExpr: x[1] + x[4] + x[5] + x[9]>
```

Corresponding Math Notation:

Define B_L to be the set of literary books.

$$\sum_{b \in B_L} x_b$$

Latex code:

```
$$ \sum_{b \in B_L} x_b $$
```

Summing consecutive indices

```
[2]: sum(x[b] for b in range(3,10))
```

```
<gurobi.LinExpr: x[3] + x[4] + x[5] + x[6] + x[7] + x[8] + x[9]>
```

**Corresponding Math Notation:

$$\sum_{b=3}^{9} x_b$$

Latex code:

```
$$\sum_{b=3}^9 x_b$$
```

Summing multiple indices

```
[3]: import pandas as pd
     cost=pd.DataFrame([[20,18,21,8],[8,23,24,8],[25,8,8,19]],\
                       index=[1,2,3],columns=['A','B','C','D'])
     FCs=cost.index
     regions=cost.columns
     y=mod.addVars(FCs,regions,name='y')
     mod.update()
     sum(cost.loc[f,r]*y[f,r] for f in FCs for r in regions )
```

```
<gurobi.LinExpr: 20.0 y[1,A] + 18.0 y[1,B] + 21.0 y[1,C] + 8.0 y[1,D] + 8.0 y[2,A] + 23.0 y[2
```

Corresponding Math Notation:

Define F to be the set of FCs and R to be the set of regions.

$$\sum_{f \in F, r \in R} c_{fr}y_{fr}$$

Latex code:

```
$$ \sum_{f \in F, r \in R} c_{fr}y_{fr} $$
```

In-Class Exercise: Writing Out the Sum Explicitly

Expand the following summations into explicit sum. You can write directly on the handout and you do not need to submit anything on Blackboard.

Example:

$S = \{1, 3, 6, 8\}$

$$\sum_{i \in S} x_i = x_1 + x_3 + x_6 + x_8$$

a)

$B = \{1, 2, 3, 4, 5, 6\}$

$$\sum_{j \in B} q_j x_j =$$

b)

$i = 3$, $J = \{2, 5, 8, 9\}$

$$\sum_{j \in J} c_{ij} y_j =$$

c)

$j = 5$

$$\sum_{i=2}^{j} a_{ij} x_i =$$

d)

$I = \{1, 2, 3\}$, $J = \{2, 4\}$, $k = 5$.

$$\sum_{i \in I, j \in J} a_{ijk} x_{ij} y_{jk} =$$

Examples of Abstract Formulation

```
[4]: # Example 1: Gurobi code from Session 19
     from gurobipy import Model, GRB
     mod=Model()
     books=range(1,11)
     booksInGenre={'Literary':[1,4,5,9],\
                   'Sci-Fi':[2,7,9],\
                   'Romance':[3,4,6,10],\
                   'Thriller':[2,3,8]}
     requirement={'Literary':2,'Sci-Fi':2,'Romance':2,'Thriller':2}
     x=mod.addVars(books, vtype=GRB.BINARY, name='x')
     mod.setObjective(sum(x[b] for b in books))
     for genre in booksInGenre:
         mod.addConstr(sum(x[b] for b in booksInGenre[genre])>=requirement[genre], name=genre)
     mod.write('10-books.lp')
     %cat 10-books.lp
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Minimize
  x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + x[8] + x[9] + x[10]
Subject To
 Literary: x[1] + x[4] + x[5] + x[9] >= 2
 Sci-Fi: x[2] + x[7] + x[9] >= 2
 Romance: x[3] + x[4] + x[6] + x[10] >= 2
 Thriller: x[2] + x[3] + x[8] >= 2
Bounds
Binaries
 x[1] x[2] x[3] x[4] x[5] x[6] x[7] x[8] x[9] x[10]
End
```

Abstract Formulation

Data:

- B: the set of books.
- G: the set of genres.
- B_g: the set of books of genre g.
- r_g: the number of books required for genre g.

Decision Variables:

- x_b: whether to carry book b. (Binary)

Objective and constraints:

$$\begin{aligned}
\text{Minimize:} && \sum_{b \in B} x_b \\
\text{subject to:} \\
\text{(Enough books in genre)} && \sum_{b \in B_g} x_b & \ge r_g & \text{ for each genre $g \in G$.}
\end{aligned}$$

Corresponding Latex

```
$$\begin{aligned}
\text{Minimize:} && \sum_{b \in B} x_b \\
\text{subject to:} \\
\text{(Enough books in genre)} && \sum_{b \in B_g} x_b & \ge r_g & \text{ for each genre $g \in G$.}
\end{aligned}$$
```

```
[5]: # Example 2: Gurobi code based on Session 19 Exercise 10.2
import pandas as pd
n=6
s=1000
months=range(1,n+1)
price=pd.Series([150,160,180,170,180,160],index=months)
demand=pd.Series([2000]*n, index=months)
mod=Model()
X=mod.addVars(months)
Y=mod.addVars(months,ub=s)
mod.setObjective(sum(price.loc[i]*X[i] for i in months))
for t in months:
    if t==1:
        mod.addConstr(Y[1]==X[1]-demand[1])
    else:
        mod.addConstr(Y[t]==X[t]+Y[t-1]-demand[t])
```

Abstract Formulation

Data:

- n: number of months.
- T: set of months. $T = \{1, 2, 3, \cdots, n\}$.
- p_t: price of oil in month t.
- d_t: demand in month t.
- s: amount of oil that can be stored at any time.

Decision Variables:

- x_t: amount of oil to buy in month t. (Continuous)
- y_t: amount of oil stored at the end of month t. (Continuous)

Objective and Constraints:

$$\begin{aligned}
\text{Minimize: } && \sum_{t \in T} p_tx_t \\
\text{s.t. } && \\
&& y_1 &= x_1 - d_1 \\
&& y_t &= x_t + y_{t-1} - d_t && \text{for each month $t \in \{2,3,\cdots,n\}$.} \\
&& y_t &\le s && \text{for each month $t \in T$.} \\
&& x_t, y_t & \ge 0
\end{aligned}$$

Corresponding Latex

```
$$\begin{aligned}
\text{Minimize: } && \sum_{t \in T} p_tx_t \\
\text{s.t. } && \\
&& y_1 &= x_1 - d_1 \\
&& y_t &= x_t + y_{t-1} - d_t && \text{for each month $t \in \{2,3,\cdots,n\}$.} \\
&& y_t &\le s && \text{for each month $t \in T$.}  \\
&& x_t, y_t & \ge 0
\end{aligned}$$
```

```
[6]: # Example 3: Gurobi code based on Session 20 Exercise 10.4
     projects=['A','B','C','D','E','F','G']
     conflicts=[['A','B'],['B','C'],['A','C'],['A','D'],\
                ['D','E'],['E','F'],['F','G'],['E','G']]
     prereqs=[['A','F'],['B','G']]
     mod=Model()
     x=mod.addVars(projects,vtype=GRB.BINARY)
     mod.setObjective(sum(x[p] for p in projects),sense=GRB.MAXIMIZE)
     for p1,p2 in conflicts:
         mod.addConstr(x[p1]+x[p2]<=1)
     for p1,p2 in prereqs:
         mod.addConstr(x[p1]>=x[p2])
```

Abstract Formulation

Data:

- P: set of projects
- C: set of conflicts. Each $(p_1, p_2) \in C$ is a pair of projects that conflicts with one another.
- R: set of prerequisite pairs. Each $(p_1, p_2) \in R$ is a pair such that project p_1 is a prerequisite to project p_2.

Decision Variables: x_p: whether to pursue project p. (Binary)
Objective and Constraints:

$$\begin{aligned}
\text{Maximize} && \sum_{p \in P} x_p \\
\text{s.t.} && \\
&& x_{p_1}+x_{p_2} &\le 1 && \text{For each conflicting pairs $(p_1,p_2) \in C$.} \\
&& x_{p_1} & \ge x_{p_2} && \text{For each pair (p_1,p_2) such that p_1 is a prereq to p_2.}
\end{aligned}$$

Corresponding Latex

```
$$\begin{aligned}
\text{Maximize} && \sum_{p \in P} x_p \\
\text{s.t.} && \\
&& x_{p_1}+x_{p_2} &\le 1 && \text{For each conflicting pairs $(p_1,p_2) \in C$.} \\
&& x_{p_2} & \le x_{p_1} && \text{For each pair $(p_1,p_2)$ such that $p_1$ is a prereq to $p_2$.}
\end{aligned}$$
```

Exercise 11.1: Abstract Formulation for Supply Chain Planning

Download the Jupyter notebook attached to the Blackboard link for this exercise and submit it there after completing it. The notebook asks you to write the abstract formulation corresponding to the following Gurobi code from Session 20.

```
[7]: # Gurobi code from Session 20
     import pandas as pd
     cost=pd.DataFrame([[20,18,21,8],[8,23,24,8],[25,8,8,19]],\
                       index=[1,2,3],columns=['A','B','C','D'])
     demand=pd.Series([30,50,10,20],index=['A','B','C','D'])
     capacity=pd.Series([40]*3, index=[1,2,3])
     FCs=cost.index
     regions=cost.columns
     mod=Model()
     x=mod.addVars(FCs,regions,name='x')
     mod.setObjective(sum(cost.loc[f,r]*x[f,r] for f in FCs for r in regions))
     for f in FCs:
         mod.addConstr(sum(x[f,r] for r in regions)<=capacity[f],name=f'Capacity_{f}')
     for r in regions:
         mod.addConstr(sum(x[f,r] for f in FCs)>=demand[r],name=f'Demand_{r}')
     mod.write('10-supplyChain.lp')
     %cat 10-supplyChain.lp
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Minimize
  20 x[1,A] + 18 x[1,B] + 21 x[1,C] + 8 x[1,D] + 8 x[2,A] + 23 x[2,B]
   + 24 x[2,C] + 8 x[2,D] + 25 x[3,A] + 8 x[3,B] + 8 x[3,C] + 19 x[3,D]
Subject To
 Capacity_1: x[1,A] + x[1,B] + x[1,C] + x[1,D] <= 40
 Capacity_2: x[2,A] + x[2,B] + x[2,C] + x[2,D] <= 40
 Capacity_3: x[3,A] + x[3,B] + x[3,C] + x[3,D] <= 40
 Demand_A: x[1,A] + x[2,A] + x[3,A] >= 30
 Demand_B: x[1,B] + x[2,B] + x[3,B] >= 50
 Demand_C: x[1,C] + x[2,C] + x[3,C] >= 10
 Demand_D: x[1,D] + x[2,D] + x[3,D] >= 20
Bounds
End
```

Abstract Formulation

Data:

Decision Variables:

Objective:

Constraints:

Exercise 11.2: Abstract Formulation for Box Selection

Download the Jupyter notebook attached to the Blackboard link for this exercise and submit it there after completing it. The notebook asks you to complete the abstract formulation corresponding to the following concrete formulation from Exercise 9.1.

Item type	1	2	3
Minimum box size (in cubit feet)	1.5	1.7	2.0
Demand	400	500	200

Concrete Formulation

Decision Variables:

- Y_1, Y_2, Y_3: how many boxes to make of each box type. (Integer)
- Z_1, Z_2, Z_3: whether to make the mold for each box type. (Binary)

Objective and Constraints:

$$
\begin{aligned}
\text{Minimize:} \quad & 1.5Y_1 + 1.7Y_2 + 2.0Y_3 + 1000(Z_1 + Z_2 + Z_3) \\
\text{s.t.} \quad & \\
\text{(Demand 1)} \quad & Y_1 + Y_2 + Y_3 \geq 1100 \\
\text{(Demand 2)} \quad & Y_2 + Y_3 \geq 700 \\
\text{(Demand 3)} \quad & Y_3 \geq 200 \\
\text{(S boxes on/off)} \quad & Y_1 \leq 1100Z_1 \\
\text{(M boxes on/off)} \quad & Y_2 \leq 1100Z_2 \\
\text{(L boxes on/off)} \quad & Y_3 \leq 1100Z_3 \\
& Y_1, Y_2, Y_3 \geq 0
\end{aligned}
$$

Abstract Formulation

Data:

- n: the number of item types, which is also equal to the number of box types.
- $I = \{1, 2, \cdots, n\}$: the set of box types, with the labels ordered in increasing box sizes.
- s_i: the size of box type i. This is also the variable cost of producing boxes of this type.
- d_i: the demand for item i.
- $M = \sum_{i=1}^{n} d_i$: the total demand of all items.
- F: the fixed cost of making each type of box. (In the concrete formulation, $F = 1000$.)

Decision Variables:

Objective and Constraints: