Paper Tree

Author: Aster Santana
July 2020

This is the documentation of the Paper Tree use case, which is part of the Learning Mip project maintained by Mip Master.

Concepts

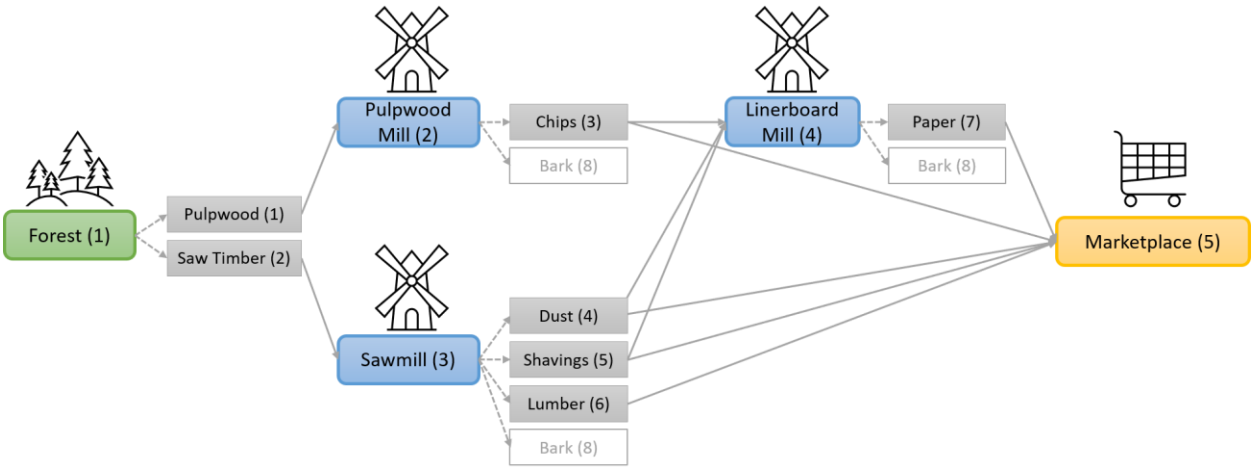
- Network flow problem
- Conservation of flow constraint
- Optimization data model

Problem statement

Mr. Mip has been contacted by Paper Tree, a company that is well-known for producing paper and other materials, such as lumber, wood ships, and pulpwood, which are all derived from the tree that they harvest.

Paper Tree is committed to invest in technology to improve their operations and wants to start from the production planning. Mr. Mip was surprised to see that, even though Paper Tree is a big company that has been in the market for a few decades now, planners have only been using spreadsheets and rules-of-thumb to plan operations across the organization.

The flow chart gives the big picture of the first problem to be solved (more complex versions to come which will involve time periods and inventory). Specifically, harvested trees are preprocessed in the land field. Larger-diameter logs yield sawtimber (also called saw logs) that go to the sawmill. Smaller-diameter logs and branches yield pulpwood and are routed to the pulpwood mill.



Three commodities come out of the sawmill: lumber, shavings, and dust. There is also a small fraction that results in bark, which is just waste and doesn't have any value. All production of lumber goes to the marketplace. Shavings and dust can either go to the marketplace or to the linerboard mill, where it's used for production of paper.

All the pulpwood that goes to the pulpwood mill becomes chips, except for a small fraction that results in bark. Chips can either go to the marketplace or to the linerboard mill for paper production. All the paper from the linerboard mill go to the marketplace.

Ultimately, Paper Tree must know how much timber and pulpwood to harvest, and how much shavings, and dust to use for paper production as to meet the demand of each commodity in the marketplace while maximizing its profit and without violating capacities.

Input data

The input data has four tables:

1. *Location Table*

Location ID	Location	Processing Cost ($/cu ft)	Processing Capacity (cu ft)
1	Forest	67.90	-
2	Pulpwood Mill	67.90	230
3	Sawmill	23.50	270
4	Linerboard	115.05	120
5	Marketplace	-	-

2. *Commodity Table*

Commodity ID	Commodity	Demand (cu ft)	Market Price ($/cu ft)
1	Pulpwood	0	-
2	Sawtimber	0	-
3	Chips	65	310.00
4	Dust	20	45.00
5	Shavings	35	64.00
6	Lumber	110	150.00
7	Paper	15	2,860.00
8	Bark	0	-

3. *Hauling Cost Table*

Origin	Destination	Commodity	Hauling Cost ($/cu ft)
Forest	Pulpwood Mill	Pulpwood	45.00
Forest	Sawmill	Sawtimber	24.00
Pulpwood Mill	Linerboard Mill	Chips	13.45
Pulpwood Mill	Marketplace	Chips	64.00
Sawmill	Linerboard Mill	Dust	11.80
Sawmill	Linerboard Mill	Shavings	11.80
Sawmill	Marketplace	Dust	75.00
Sawmill	Marketplace	Shavings	75.00
Sawmill	Marketplace	Lumber	62.50
Linerboard Mill	Marketplace	Paper	44.05

4. *Output Ratio Table*

Location	Material	Product	Output Ratio
Pulpwood Mill	Pulpwood	Chips	0.85
Pulpwood Mill	Pulpwood	Bark	0.15
Sawmill	Sawtimber	Dust	0.10
Sawmill	Sawtimber	Shavings	0.15
Sawmill	Sawtimber	Lumber	0.70
Sawmill	Sawtimber	Bark	0.05
Linerboard Mill	Chips	Paper	0.25
Linerboard Mill	Chips	Bark	0.75
Linerboard Mill	Dust	Paper	0.55
Linerboard Mill	Dust	Bark	0.45
Linerboard Mill	Shavings	Paper	0.40
Linerboard Mill	Shavings	Bark	0.60

Formulation

Although the problem might seem a bit complex at first, Mr. Mip immediately realized that this a *network flow problem*: there is a graph (in which nodes are locations/facilities and arcs are transportation lanes) and the goal is to decide how much commodity flows along each arc given costs, capacities, and demands.

Knowing that Paper Tree will need to solve this problem recurrently as the data change, Mr. Mip decided to write a generic formulation.

By generic formulation Mr. Mip means defining an *input data model* to write the formulation, opposed to using actual data. Then, once the data is available, he populates this data model to solve the optimization as many times as needed.

Input data model

Mr. Mip always design the *optimization input data model* in two steps:

1. Define the set of indices
2. Define the parameters (as a function of the indices)

Set of indices

I Set of locations.

K Set of commodities.

Elements of I are denoted by i and j, and elements of K are denoted by k.

Parameters

pc_i Processing cost ($/cu ft) over the input at location i.

pu_i Processing upper bound /capacity (cu ft) at location i.

hc_{ijk} Hauling cost ($/cu ft) from location i to location j, of commodity k.

d_k Demand (cu ft) of commodity k in the marketplace.

mp_k Market price ($/cu ft) of commodity k in the marketplace.

r_{ij} Output ratio of product j from processing material i.

To keep the formulation and the code clean, Mr. Mip always uses IDs instead of actual names. In this case, IDs are defined for locations and commodity in Location Table and Commodity Table. Using IDs is also recommended for performance reasons, since lengthy strings can hurt the performance of the optimization when solving large models.

Systematic as he is, Mr. Mip has some conventions when it comes to define the optimization input data model (which you may have observed in this example):

- Set of indices are always denoted with capital letters, usually I, J, K, L, and T. And its elements are always denoted with lowercase, like $i \in I$ and $t \in T$.
- Parameters are always denoted with lowercase letters. Suggestive names are preferred but they should be very short, one or two letters, to keep the formulation and the code concise and readable.
- The implementation code uses exactly the same notation adopted in the data model. Although the code is written in a programming language, think of the code as the formulation typed in a different format. The more alike they are, the easier to maintain.

Decision variables

Although it may seem like there are lots of different type of decisions to make, Mr. Mip knows that it's all about the amount of flow that goes along each arc of the network. Therefore, he only defines one set continuous variables, the flow variables.

Decision variables:

x_{ijk} Flow (cu ft), from location i to j, of commodity k.

OBS: Not all (i, j, k)-combinations are feasible. For example, there is no paper, $k = 7$, going from the forest, $i = 1$, to the sawmill, $j = 3$. In fact, all feasible combinations are readily available in the Hauling Cost Table. There are only 10 of them. Some people like to define variables for all possible combinations, $5 \times 5 \times 8 = 200$ in this case, and them "kill" these infeasible variables by setting their upper bound to be zero in the model. Mr. Mip rolls his eyes most of the times he sees this. There are only a few cases where it makes sense to add infeasible variables to the model and force them to zero.

Constraints

There are only three types of constraints for this problem: capacity, demand, and flow balance constraints.

There are tree capacity constraints, one for each mill.

Constraint – Input cannot exceed the processing capacity of the pulpwood mill:

$$x_{121} \leq pu_2.$$

Constraint – Input cannot exceed the processing capacity of the sawmill:

$$x_{132} \leq pu_3.$$

Constraint – Input cannot exceed the processing capacity of the linerboard mill:

$$x_{243} + x_{344} + x_{345} \leq pu_4.$$

There are typically two types of demand constraints, the one that demand must be met exactly and the one that allows for excess demand. In the case of Paper Tree, excess demand is allowed only for dust, shavings, and lumber.

Constraint – Demand for lumber must be met at least:

$$x_{356} \geq d_6.$$

Constraint – Demand for shavings must be met at least:

$$x_{355} \geq d_5.$$

Constraint – Demand for dust must be met at least:

$$x_{354} \geq d_4.$$

Constraint – Demand for chips must be met exactly:

$$x_{253} = d_3.$$

Constraint – Demand for paper must be met exactly:

$$x_{457} = d_7.$$

Finally, there are the flow balance constraints. You will see flow balance constraints in every flow problem. They model the fact that flow-in must equal flow-out for every commodity at every node except, possibly, at supply and demand nodes.

This problem has five nodes. Node 1, the forest, is a supply node. Node 5, the marketplace, is a demand node. Nodes 2, 3, and 4 are called intermediate nodes.

Mr. Mip has taken care of Node 5 with the demand constraints, which are a kind of flow balance constraints. When he asked about the availability of pulpwood and sawtimber from the forest, Paper Tree said to assume for now that there is sufficient supply. Therefore, no constraint is needed for Node 1.

Mr. Mip also asked about the destination of bark. Paper Tree said that they let it accumulate in the mill for several weeks and only when there is a significant amount, they get rid of it.

Considering the output ratios of chips from pulpwood, Mr. Mip arrived at the following flow balance constraint for the pulpwood mill.

Constraint – Flow balance of chips at the pulpwood mill:

$$r_{13}x_{121} = x_{243} + x_{253}.$$

Mr. Mip didn't write a similar constraint for bark because the is no need to track bark.

Flow balance constraint for dust, shavings, and lumber at the sawmill are similar.

Constraint – Flow balance of dust at the sawmill:

$$r_{24}x_{132} = x_{344} + x_{354}.$$

Constraint – Flow balance of shavings at the sawmill:

$$r_{25}x_{132} = x_{345} + x_{355}.$$

Constraint – Flow balance of lumber at the sawmill:

$$r_{26}x_{132} = x_{356}.$$

Flow balance constraint at the linerboard mill is also similar, except that there are three materials coming in, and from two different locations.

Constraint – Flow balance of paper at the linerboard mill:

$$r_{37}x_{243} + r_{47}x_{344} + r_{57}x_{345} = x_{457}.$$

Objective

The objective is to maximize profit, i.e., revenue minus cost.

The total revenue is just the sum of the flows of each commodity into the market multiplied by the respective price. For example, the revenue from chips can be formulated as mp_3x_{253}.

Therefore, Mr. Mip obtained the total revenue as follows:

$$revenue = mp_3x_{253} + mp_4x_{354} + mp_5x_{355} + mp_6x_{356} + mp_6x_{356}.$$

The total processing cost is the sum of the processing cost from all the locations.

The cost from processing sawtimber and pulpwood in the forest is $pc_1(x_{121} + x_{132})$, where $x_{121} + x_{132}$ is the total output from the forest. Mr. Mip used the output because there is no input to the forest. But for all the other locations, the processing cost is over the total input. For example, the processing cost at the linerboard mill is $pc_4(x_{243} + x_{344} + x_{345})$.

Therefore, Mr. Mip obtained the total revenue as follows:

$$processingCost = pc_1(x_{121} + x_{132}) + pc_2x_{121} + pc_3x_{132} + pc_4(x_{243} + x_{344} + x_{345}).$$

Finally, the hauling cost is just the sum of the hauling cost from each arc. For example, the hauling cost from the sawmill to the linerboard mill is given by the flow of dust and shavings on that arc multiplied by the respective costs: $hc_{344}x_{344} + hc_{345}x_{345}$.

Therefore, Mr. Mip obtained the total revenue as follows:

$$haulingCost = \sum_{i,j,k} hc_{ijk}x_{ijk}.$$

Putting them all together, the final objective function becomes:

Objective:

$$\max\ (revenue - processingCost - haulingCost).$$

Final formulation

Putting all the pieces together, Mr. Mip arrived at the following formulation:

Final formulation:

$$\begin{aligned}
\max \quad & revenue - processingCost - haulingCost \\
s.t. \quad & \\
& x_{121} \leq pu_2 && \text{cap. pulpwood mill} \\
& x_{132} \leq pu_3 && \text{cap. sawmill} \\
& x_{243} + x_{344} + x_{345} \leq pu_4 && \text{cap. linerboard mill} \\
& x_{253} = d_3 && \text{demand chips} \\
& x_{354} \geq d_4 && \text{demand dust} \\
& x_{355} \geq d_5 && \text{demand shavings} \\
& x_{356} \geq d_6 && \text{demand lumber} \\
& x_{457} = d_7 && \text{demand paper} \\
& r_{13}x_{121} = x_{243} + x_{253} && \text{flow bal. chips} \\
& r_{24}x_{132} = x_{344} + x_{354} && \text{flow bal. dust} \\
& r_{25}x_{132} = x_{345} + x_{355} && \text{flow bal. shavings} \\
& r_{26}x_{132} = x_{356} && \text{flow bal. lumber} \\
& r_{37}x_{243} + r_{47}x_{344} + r_{57}x_{345} = x_{457} && \text{flow bal. paper} \\
& x_{ijk} \geq 0 \text{ for all } i, j, k.
\end{aligned}$$

Where,

$$revenue = mp_3x_{253} + mp_4x_{354} + mp_5x_{355} + mp_6x_{356} + mp_6x_{356}$$

$$processingCost = pc_1(x_{121} + x_{132}) + pc_2x_{121} + pc_3x_{132} + pc_4(x_{243} + x_{344} + x_{345})$$

$$haulingCost = \sum_{i,j,k} hc_{ijk}x_{ijk}.$$

Implementation and optimization

Next is an implementation of this formulation. In this version, the data parameters are hard-coded. Mr. Mip's version reads the data directly from csv files, which is recommended. The code is available in the Mip Master repository on GitHub [link].

In the first block has only the hard-coded data, which uses the same notation defined in the input data model section.

```
1  import gurobipy as gp
2
3  # Input Data
4  # locations
5  I = {1: 'Forest', 2: 'Pulpwood Mill', 3: 'Sawmill',
6       4: 'Linerboard Mill', 5: 'Marketplace'}
7  # commodities
8  K = {1: 'Pulpwood', 2: 'Sawtimber', 3: 'Chips', 4: 'Dust',
9       5: 'Shavings', 6: 'Lumber', 7: 'Paper', 8: 'Bark'}
10 # processing cost
11 pc = {1: 67.9, 2: 67.9, 3: 23.5, 4: 115.05, 5: 0.0}
12 # production upper bound/capacity
13 pu = {2: 230.0, 3: 270.0, 4: 120.0}
14 # hauling cost
15 hc = {(1, 2, 1): 45.0, (1, 3, 2): 24.0, (2, 4, 3): 13.45,
16       (2, 5, 3): 64.0, (3, 4, 4): 11.8, (3, 4, 5): 11.8,
17       (3, 5, 4): 75.0, (3, 5, 5): 75.0, (3, 5, 6): 62.5,
18       (4, 5, 7): 44.05}
19 # demand
20 d = {3: 65, 4: 20, 5: 35, 6: 110, 7: 15}
21 # market price
22 mp = {3: 310, 4: 45, 5: 64, 6: 150, 7: 2860}
23 # output ratio
24 r = {(1, 3): 0.85, (1, 8): 0.15, (2, 4): 0.1, (2, 5): 0.15,
25      (2, 6): 0.7, (2, 8): 0.05, (3, 7): 0.25, (3, 8): 0.75,
26      (4, 7): 0.55, (4, 8): 0.45, (5, 7): 0.4, (5, 8): 0.6}
27 # variables keys
28 x_keys = list(hc.keys())
29
```

Two observations here:

1. Mr. Mip usually define the set of indices, I and K in this case, as lists and use them to write constraints. However, in this problem, constraints are added one at a time. Therefore, we used I and K to map locations and commodities ID's to its names.
2. In Line 28, Mr. Mip explicitly define the list of tuples that will define the variables keys. This is a good practice for two reasons. First, it helps preventing key errors when building expressions to add constraints and the objective function. Second, it avoids defining unnecessary variables as mentioned before. For example, someone might define

   ```
   x = mdl.addVars(I, I, K, vtype=gp.GRB.CONTINUOUS, name='x').
   ```

   Which is equivalent to define

   ```
   x_keys = [(i, j, k) for i in I for j in J for k in K].
   ```

Obviously, many of the (i, j, k)-combinations defined this way would be infeasible because not all these arcs are part of the actual network and not all commodities flow through every arc.

In the second block of code is the optimization model.

```
# Define the model
mdl = gp.Model('PaperTree')

# Add variables
x = mdl.addVars(x_keys, vtype=gp.GRB.CONTINUOUS, name='x')

# Add Constraints
# capacity
mdl.addConstr(x[1, 2, 1] <= pu[2], name='cap_pulpwood_mill')
mdl.addConstr(x[1, 3, 2] <= pu[3], name='cap_sawmill')
mdl.addConstr(x[2, 4, 3] + x[3, 4, 4] + x[3, 4, 5] <= pu[4],
              name='cap_linerboard_mill')
# demand
mdl.addConstr(x[2, 5, 3] == d[3], name='demand_chips')
mdl.addConstr(x[3, 5, 4] >= d[4], name='demand_dust')
mdl.addConstr(x[3, 5, 5] >= d[5], name='demand_shavings')
mdl.addConstr(x[3, 5, 6] >= d[6], name='demand_lumber')
mdl.addConstr(x[4, 5, 7] == d[7], name='demand_paper')
# flow balance
mdl.addConstr(r[1, 3] * x[1, 2, 1] == x[2, 4, 3] + x[2, 5, 3],
              name='flow_bal_chips')
mdl.addConstr(r[2, 4] * x[1, 3, 2] == x[3, 4, 4] + x[3, 5, 4],
              name='flow_bal_dust')
mdl.addConstr(r[2, 5] * x[1, 3, 2] == x[3, 4, 5] + x[3, 5, 5],
              name='flow_bal_shavings')
mdl.addConstr(r[2, 6] * x[1, 3, 2] == x[3, 5, 6],
              name='flow_bal_lumber')
mdl.addConstr(r[3, 7] * x[2, 4, 3] + r[4, 7] * x[3, 4, 4] +
              r[5, 7] * x[3, 4, 5] == x[4, 5, 7],
              name='flow_bal_paper')

# Set the objective function
revenue = sum(mp[k] * x[i, j, k] for (i, j, k) in x_keys if j == 5)
hauling_cost = sum(hc[key] * x[key] for key in x_keys)
processing_cost = pc[1] * (x[1, 2, 1] + x[1, 3, 2]) + \
                  pc[2] * x[1, 2, 1] + pc[3] * x[1, 3, 2] + \
                  pc[4] * (x[2, 4, 3] + x[3, 4, 4] + x[3, 4, 5])
mdl.setObjective(revenue - processing_cost - hauling_cost,
                 sense=gp.GRB.MAXIMIZE)

# Optimize
mdl.optimize()

# Retrieve the solution
print(f'Total profit {mdl.objVal:.2f}')
for (i, j, k), v in x.items():
    print(f'From {I[i]:<15} to {I[j]:<15} {v.X:5.2f} {K[k]}')
```

Observe how the x_keys makes it easy to define the total revenue and hauling cost in Line 62-63. In Line 76, the dictionaries I and K are used to print the name of locations and commodities instead of its ID. The output is this:

```
Total profit 14997.07
From Forest          to Pulpwood Mill   118.59 Pulpwood
From Forest          to Sawmill         270.00 Sawtimber
From Pulpwood Mill   to Linerboard Mill 35.80 Chips
From Pulpwood Mill   to Marketplace     65.00 Chips
From Sawmill         to Linerboard Mill  7.00 Dust
From Sawmill         to Linerboard Mill  5.50 Shavings
From Sawmill         to Marketplace     20.00 Dust
From Sawmill         to Marketplace     35.00 Shavings
From Sawmill         to Marketplace     189.00 Lumber
From Linerboard Mill to Marketplace     15.00 Paper
```

Of course, Mr. Mip exports this output to a csv file rather than just printing it out to the console.

Challenge yourself

1. A local company called Four Seasons contacted Paper Tree proposing a better solution for dealing with the bark. Every night, Four Seasons will take all the bark produced in the three mills and only charge the transportation cost, because they use the bark to produce natural fertilizers and sell it to the marketplace.
   The transportation cost for Four Seasons to haul bark from the pulpwood mill, sawmill, and linerboard mill to their facility, is $8.50, $9.00, and $7.20 per cubic feet, respectively.
   Happy to know that the bark they produce will have an environment friendly destination, Paper Tree closed the deal with Four Season. It's your job now to add this new business requirement to Mr. Mip's model. If you need some help, there is a hint in the footnote[1].
   After optimizing the new model, you should see the total profit dropping to $14,484.61.

Takeaways

1. For formulations that requires several parameters, it's a good practice to design an optimization data model to write the formulation. The data model should have two sections: set of indices denoted and parameters. By following Mr. Mip's standards you will have a clean formulation and neat implementation.
2. Network flow problems are common, and conservation of flow constraints are always part of the formulation. If there are multiple commodities, one flow balance constraint is likely needed for each commodity and each node through which the commodity flows.
3. It's a good practice to add to the model only decision variables that are feasible. And it's a very good idea to define beforehand the list of tuples of all the decision variables of the model.

[1] You need to modify the parameters I, hc, and mp, and add three new flow balance constraints.