Logistics 2020/2021

Load Company

Marianna Abbattista(621809)[1], Rossella Bartaloni(563589)[2], and Fabio Michele Russo(429892)[3]

Contacts: [1]m.abbattista1@studenti.unipi.it, [2]r.bartaloni@studenti.unipi.it, [3]f.russo11@studenti.unipi.it

January 23, 2021

1 Description of Load Company

The aim of this project is to prepare a loading plan for a freighter, so that *Load Company* would be able to transport agricultural commodities for a specific dealer from Newport News, Virginia to a port in Ghana. The **commodities** to load are:

Commodity	Amount Available (tons)	Volume por Ton (cubic feet)	Profit per Ton ($)
1	4800	40	70
2	2500	25	50
3	1200	60	60
4	1700	55	80

Table 1: Commodities specifics

Additionally, the cargo ship used for transport has three cargo holds, with **capacities**:

Cargo Hold	Weight Capacity (tons)	Volume Capacity (cubic feet)
1 (Forward)	3000	145000
2 (Center)	6000	180000
3 (Rear)	4000	155000

Table 2: Cargoes specifics

Furthermore, our problem introduces additional limitations due to the fact that the company can place **only one type** of commodity in each cargo hold. Finally, the cargo holds also have some **balance** restrictions: the weight in the forward cargo must be within 10% of the weight in the rear. The center cargo has to hold between 40% and 60% of the entire weight loaded on the ship. Our objective is to **maximize** the total profit for the Company while complying with all the constraints. Here is a graphical representation of the problem:

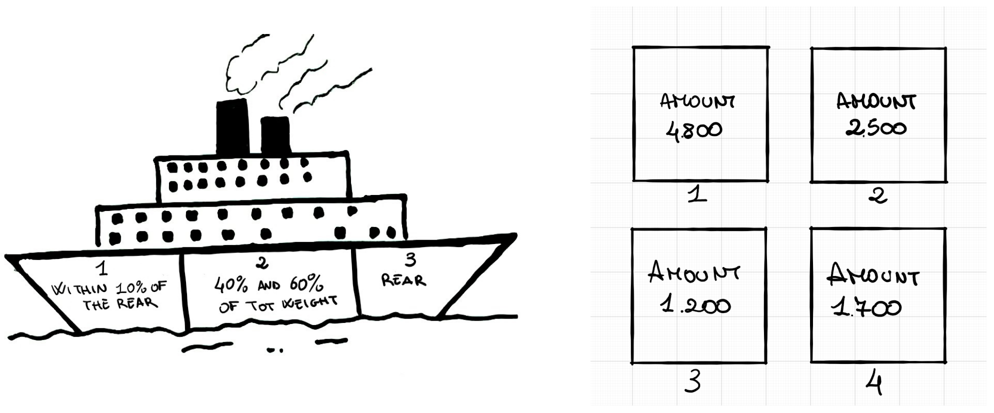


Figure 1: Representation of the problem with the three cargoes on the left and the four commodities on the right.

2 Mathematical Model

2.1 Integer Linear Programming Model

Here we present our ILP model for the problem, with the sets of commodities I and cargoes J so formed:

$$I = \{1, 2, 3, 4\};\ J = \{1, 2, 3\}$$

Objective Function

$$max \left(70 \sum_{j=1}^{3} y_{1j} + 50 \sum_{j=1}^{3} y_{2j} + 60 \sum_{j=1}^{3} y_{3j} + 80 \sum_{j=1}^{3} y_{4j} \right)$$

One-per-cargo

$$\sum_{i=1}^{4} x_{ij} = 1 \qquad \forall j \in J$$

Linking Constraint

$$M x_{ij} \geq y_{ij} \qquad \forall i \in I; \forall j \in J$$

Capacity Constraints

$$\sum_{i=1}^{4} y_{ij} \leq WeightCapacity_j \qquad \forall j \in J$$

$$\sum_{i=1}^{4} (y_{ij} \cdot Volume_i) \leq VolumeCapacity_j \qquad \forall j \in J$$

Availability Constraint

$$\sum_{j=1}^{3} y_{ij} \leq Availability_i \qquad \forall i \in I$$

Balance Constraints

$$0.9 \sum_{i=1}^{4} y_{i3} \leq \sum_{i=1}^{4} y_{i1} \leq 1.1 \sum_{i=1}^{4} y_{i3}$$

$$0.4 \cdot \left[\sum_{j=1}^{3} \sum_{i=1}^{4} y_{ij} \right] \leq \sum_{i=1}^{4} y_{i2} \leq \left[\sum_{j=1}^{3} \sum_{i=1}^{4} y_{ij} \right] \cdot 0.6$$

Integrality, Non-Negativity Constraint

$$y_{ij} \geq 0, integer \qquad \forall i \in I; \forall j \in J$$

Binary Constraint

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I; \forall j \in J$$

2.2 Description

In this problem we are dealing with two *sets*:

1. our 4 **commodities**, which are $i \in I = \{1, 2, 3, 4\}$, and
2. the 3 parts of the ship, which we have called **cargoes** and are $j \in J = \{1, 2, 3\}$.

As our decision is the quantity to load of each commodity *and* in which cargo to load each quantity, we have chosen to set two **decision variables**. These are:

1. y_{ij}, the quantity in tons to load of each commodity i on each cargo j;
2. x_{ij}, a binary variable that is 1 if the commodity i is to be loaded on cargo j, 0 otherwise.

Clearly the most complete decision variable is y_{ij}. The team that has to load the ship can get the **complete solution** of the problem by only looking at y_{ij}. However, the inclusion of x_{ij} lets us very easily add one of the constraints; that is, that we can only load one commodity per cargo.
Our **objective function** is the maximisation of the profit. That is expressed as the summation of the *profit for a commodity* times the *quantity loaded over the entire ship of that commodity.* Only the variable y_{ij} appears in this function.

2.2.1 Linking and One-per-cargo Constraints

- To **link** the variables x_{ij} to y_{ij} we have introduced that y_{ij} must not exceed $M * x_{ij}$. This boils down to one of two cases:

  1. If y_{ij} is set to 1 or more, x_{ij} *must* be set more than 0 (therefore to 1 as it is binary).
  2. On the other hand if y_{ij} is set to 0, the CPLEX solver does not need to set x_{ij} to more than 1.

  Does this mean that it will set it to 0? Theoretically, no. However, all we need from our model is that if we load commodity i in cargo j our x_{ij} reflect this fact by being assigned as 1. This will happen for sure and that is all we need for the following one-per-cargo constraint. If the commodity is not loaded on that cargo, for x_{ij} the solver will have the option of setting it to 0 or 1, indifferently. However, in case it tries to set it as 1 and that collides with the 1-per-cargo constraint, it will afterwards set it to 0 eliminating the problem.

- The **One-per-cargo constraint** is that the summation of all our x_{ij} for one cargo is exactly equal to 1, for each cargo. This means that exactly one commodity will be loaded on each of the three cargoes. This in general is different than the problem requirement (that is, that no more than one commodity must be loaded on each cargo) but for this case and given our objective function this constraint is equivalent. These are the reasons:

  1. Since we need to maximize the profit, and since there is no cost to "activate" a cargo or use it, having a cargo completely empty when we could load something on it makes no sense. Loading a commodity in an empty cargo will always lead to a better profit.
  2. We cannot load more than 1 commodity on each cargo due to problem definition.

  Having considered these, the constraint is set to exactly one commodity in each cargo.

  Note that if we change the parameters this constraint might fail. For instance, if we had only 2 commodities available to be loaded, clearly we could not load one commodity on each of the three cargoes, so this constraint would block the finding of an optimal viable solution, which would otherwise exist. (We have inserted a **check** in the AMPL model to ensure this is respected.) However, with our data, this is fine.

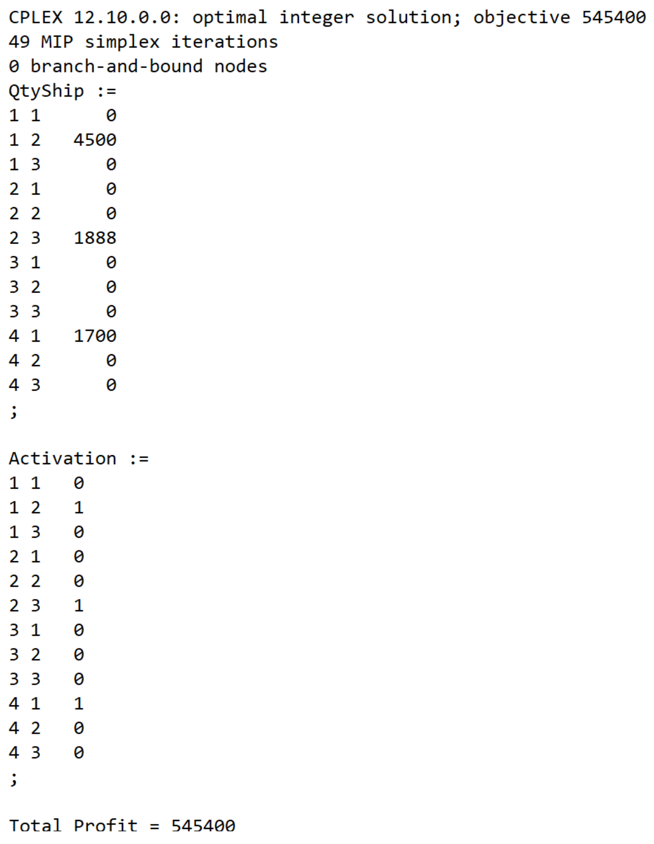
2.2.2 Other Constraints

- **Weight capacity constraint**: that the sum (in tons) of all that's loaded on a cargo must not exceed its weight capacity (for each cargo).
- **Volume capacity constraint**: that the sum for all commodities of the tons of a commodity loaded on a cargo multiplied by the commodity volume per ton must not exceed the cargo's volume capacity (for each cargo).
- **Availability constraint**: that the quantity loaded of a commodity over all the cargoes cannot exceed the quantity available of that commodity (for each commodity).
- **1st balance constraint**: that the weight loaded in cargo 1 (the forward cargo) must be between 0.9 times the weight loaded in cargo 3 and 1.1 times the weight loaded in cargo 3 (the back cargo).
- **2nd balance constraint**: that the weight loaded in cargo 2 (the central cargo) must be between 0.4 and 0.6 of the entire weight loaded on the ship.

3 CPLEX Results

After the **AMPL** implementation (that we are attaching to this report) the value of the objective function from the **CPLEX** solver is **Total profit = 545400** and the optimal solution is:

- **Commodity 1**: 4500 tons in **Cargo 2**,
- **Commodity 2**: 1888 tons in **Cargo 3**,
- **Commodity 3**: none loaded,
- **Commodity 4**: 1700 tons in **Cargo 1**.



```
CPLEX 12.10.0.0: optimal integer solution; objective 545400
49 MIP simplex iterations
0 branch-and-bound nodes
QtyShip :=
1 1      0
1 2   4500
1 3      0
2 1      0
2 2      0
2 3   1888
3 1      0
3 2      0
3 3      0
4 1   1700
4 2      0
4 3      0
;

Activation :=
1 1   0
1 2   1
1 3   0
2 1   0
2 2   0
2 3   1
3 1   0
3 2   0
3 3   0
4 1   1
4 2   0
4 3   0
;

Total Profit = 545400
```

Figure 2: CPLEX output

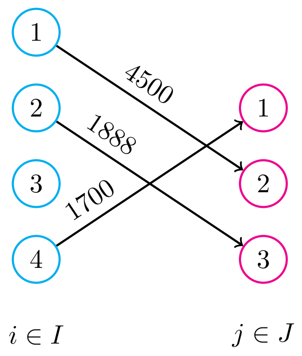


Figure 3: Bipartite graph of the problem solution with the quantity of the commodities allocated

4 Additional Constraint

The problem introduces an additional constraint: if some tons of product 1 are loaded then some tons of products 2 also need to be loaded. The model is identical as before but with one additional **conditional loading constraint**, which follows.

4.1 Addition to the mathematical model

Conditional loading constraint

$$\sum_{j=1}^{3} x_{2j} \geq \sum_{j=1}^{3} x_{1j}$$

The sum over all cargoes of x_{2j} is 1 if commodity 2 is loaded, 0 if not. Same thing for x_{1j} and commodity 1. Like for the linking constraint, there are two scenarios:

1. If $\sum_{j=1}^{3} x_{1j}$ is 1, then $\sum_{j=1}^{3} x_{2j}$ must also be set to at least 1 (i.e., with our specific data, exactly 1 because for profit maximization purposes it won't make sense to load the same commodity over different cargoes with the stated available quantities. This might not be true with different data, particularly the availabilities).
2. On the other hand, if $\sum_{j=1}^{3} x_{1j}$ is 0, $\sum_{j=1}^{3} x_{2j}$ can be any value (i.e. will be either 0 or 1 for the same reasons).

4.2 CPLEX results

For this last part, after the implementation with AMPL we had the same results as the previous one, and that is because we already have commodities 2 and 1 in our solution. So, also after having inserted the new constraint the solution is not changed, since the two commodities were already loaded. In the following there are the commodities that we selected, the amount and the cargo to which they were allocated:

- **Commodity 1**: 4500 tons in **Cargo 2**,
- **Commodity 2**: 1888 tons in **Cargo 3**,
- **Commodity 3**: none loaded,
- **Commodity 4**: 1700 tons in **Cargo 1**.

Finally, the **Total Profit = 545400**, same as before.