Optimization Techniques for Logistical Tendering

Skylar Carpenter
Ayotunde Egunjobi
Katie Heidt
Justin Reising

Fall 2018

Abstract

Logistical tendering is a common business practice among companies for which distribution of products are in increasingly globalized markets. The ability to make decisions surrounding the optimization of complex supply chains quickly and efficiently can have significant effects on a company's market position. The reverse auction format consists of a number of transportation lanes for which distribution services are required and bids from multiple vendors on these transportation lanes. This work will focus primarily on the extension of Jennifer Kiser's thesis [2] with an updated list of vendors and transportation lanes from our industry liaison for solving the optimization problem commonly referred to as the "Winner Determination Problem" with added complexities and constraints.

1 Introduction

1.1 Background

For any company that produces goods, a major consideration is the distribution of those goods. Depending on the size of the business, this may range from local distribution with just a few destinations, to worldwide with hundreds of destinations. Large companies who are shipping products to and from many different locations may choose to use third-party distributors. A process known as tendering is used by many companies to select these third-party distributors to transport their goods [1]. A tender is usually established through a bidding process.

In a traditional auction, the individual placing the highest bid is the winner and agrees to pay the bid amount for the goods or service. In a reverse auction, however, the seller is offering buyers the opportunity to perform a service, and so here the lowest bid is the winner, given no further constraints. In both of these scenarios, bids are only for single items or services.

1.2 Motivation

From the perspective of a company who produces goods and uses tendering to select distributors, there will likely be multiple distribution lanes that need to be covered. In the simplest scenario, the company (also referred to as the buyer or shipper) will want to select the distributors (also known as the vendor, supplier, or carrier) that minimizes their cost while ensuring that each lane is covered. The buyer will provide potential bidders with information on the expected demand for each lane, and the interested suppliers will respond with a price (bid) for which they would complete the job.

However, there may be other constraints that the company wishes to meet. These may include but are certainly not limited to: choosing to work with specific distributors, only working with a set maximum or minimum number of distributors, or selecting distributors who are known to complete their transports quicker [1]. The company must also consider the capabilities of the distributors; most will have a limit to the volume that they can transport in a given time or to a certain destination. Bundling of bids allows vendors to bid on lanes that maximize their efficiency by considering not only the delivery but also the return trip. This scenario sets up what is known as an optimization problem: the company wants to find a solution to a cost function subject to specific constraints [1], that is, they wish to minimize the total cost for transporting their goods while meeting the necessary conditions.

1.3 Preliminary Data Analysis

The data with which we are working comes from Eastman Chemical Company, located in Kingsport, TN. There are 16 carriers (distributors) in this data set, 9 of which are identified as local and 7 as global. A total of 2199 bids were placed by these carriers for the 234 lots (lanes) available. Tables 1 and 2 give a preview of the data.

Table 1: A Sample of the Lot Data.

Lot #	**# Shipments**	**As/Is Cost/Shipment**	**Site Cluster**	**Incumbent**	**Lowest Bid**	**# Bids**
1	3	2686.8	Benelux	VTGT	2864.4	11
2	3	2362.8	Benelux	INTT	2337.6	11
3	2	3092.4	Benelux	MSLO	2456.4	8
4	1	2755.2	Benelux	MSLO	2214.0	5
5	6	2624.4	Benelux	BLKH	2143.2	12

Table 1 gives information about the 234 lots (lanes) on which the carriers were bidding. This data includes the lot number, the number of shipments per lot, the as/is cost/shipment (which is the cost from last year), the site cluster (identifying to which of the 4 clusters the lane belongs), the incumbent carrier (based on last year's data, the accepted bid), and the total number of bids.

Table 2: A Sample of the Bid Data.

Bid	**Carrier**	**Lot**	**Cost**	**Rank**	**Lowest Bid**	**Delta vs Lowest**	**# Shipments/Lot**
1	INTT	ISO_001	2686.4	1	2864.4	0	3
2	ITLW	ISO_001	2970.0	2	2864.4	105.6	3
3	VTGT	ISO_001	3335.6	3	2864.4	361.2	3
4	LXSG	ISO_001	3499.2	4	2864.4	634.8	3
5	HOGT	ISO_001	3915.6	5	2864.4	1051.2	3

Table 2 gives information about the 2199 bids that were placed. This data includes the bid number, the carrier who placed the bid, the lot number, the cost (which is the value of the bid), the rank for each bid in each lane (with 1 being the cheapest bid for that lane), the lowest placed bid for that lane, delta vs lowest represents the difference of each of the other bids in a given lane from the lowest, and the number of shipments per lot.

Before attempting to create the cost function and identifying the constraints, we first perform some preliminary analysis. The following histogram (Figure 1) displays the frequency of bids per lane. From this, we can see that most lanes have 8 to 12 bids, while the minimum number of bids for any lane was 2 and the maximum was 15. Additionally, we found that the average number of bids was 9.4 and the median was 10, so this distribution is slightly left skewed.

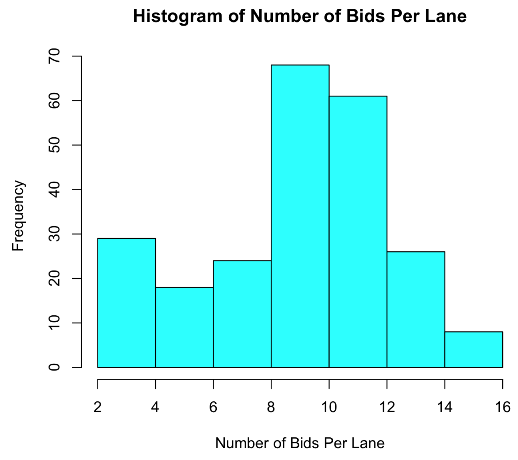


Figure 1: Histogram of number of bids per lane.

Figure 2 below shows the distribution of the number of shipments for the 234 lanes. We can see that this is a severely right-skewed distribution, as the median number of shipments is only 3 and the mean is 15. There are several outliers in this portion of the data, with the maximum being 470 shipments.

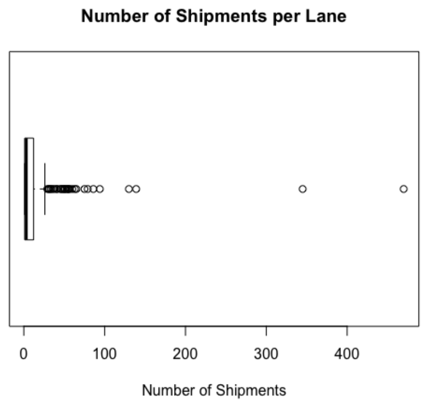


Figure 2: Boxplot of the number of shipments per lane.

Next, we consider the distribution of bids placed for three of the lanes that had 15 bids, making them some of the more competitive lanes in the tender. In the boxplot below (Figure 3), we can see the distribution of bids for these lanes; the bids for lane 47 range from \$780 to \$1800, lane 48 ranges from \$1080 to \$2388, and lane 137 ranges from \$1080 to \$2250. This gives us an idea of the amount vendors are bidding.

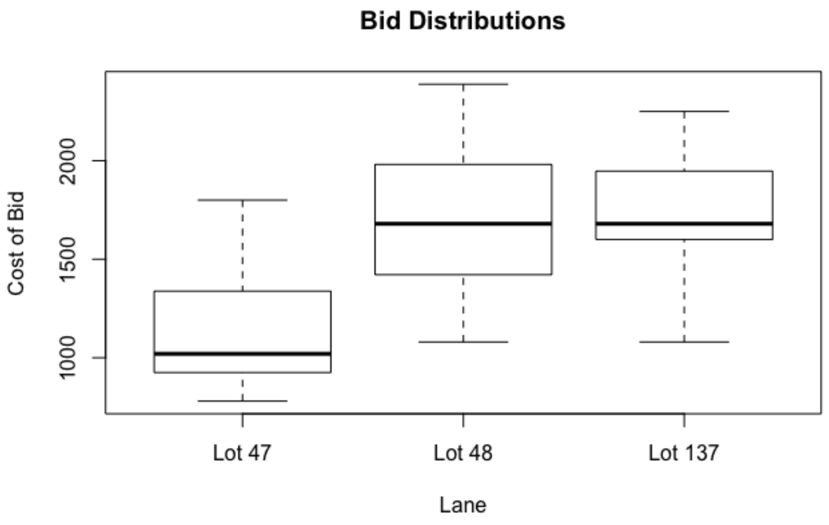


Figure 3: Side by side boxplots of the bids for lanes 47, 48, and 137.

If the objective were simply to find the lowest bid for each lane, then of course the bids of \$780, \$1080, and \$1080 would be chosen. However, our goal is to find a cost function that minimizes the total cost of transportation subject to the constraints given. These constraints may require that certain suppliers be chosen even if they are more expensive, that only a maximum number of suppliers are used, or that the total volume assigned to each carrier not exceed their capabilities. It is these constraints that make the optimization problem more complex than simply choosing the lowest bids.

Lastly, consider the barplot in Figure 4 detailing the number of bids placed by each of the 16 carriers. We can see that some of the carriers placed over 200 bids, while others placed fewer than 50. This will be of interest to us later as we investigate which carriers actually win bids.

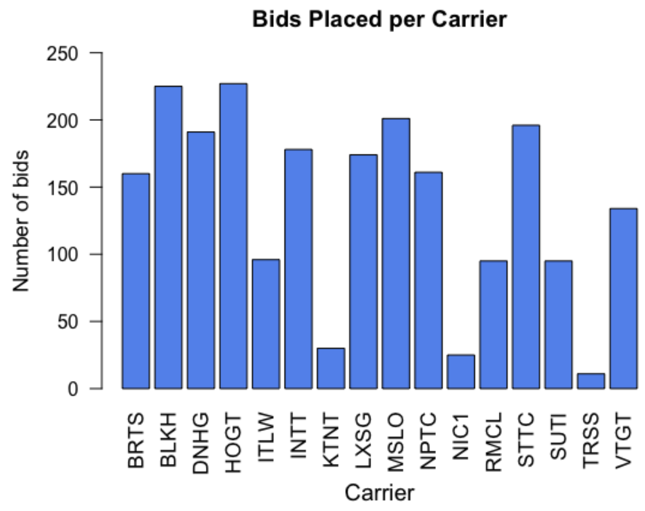


Figure 4: Barplot of bids placed per carrier.

2 Cost Minimization Model

2.1 Objective

The objective for the cost minimization model, also referred to as our base model, is to construct an objective function that minimizes the cost of the set of expected bids. In this project, we first recreated the base optimization linear programming model in [2] with slight modifications to adjust for the data set provided. We include constraints to ensure that a feasible solution is obtained. This feasible solution must provide a set of winning bids that meet the buyer's demand, meaning the number of shipments across all lanes must be covered. In the cost minimization model, each bid must be accepted in an all-or-nothing fashion, each lane cannot be awarded more than one time, and all shipments on a particular lane must be awarded to a single carrier.

2.2 Model Construction in Pyomo

In order to construct the cost minimization model, four steps were implemented in Pyomo:

1. Data was imported with each variable named as described by the mathematical equations (given in Section 2.3).

2. The objective function was stated with the constraints.

3. Demand on each lane was initially set to one to ensure that only one bid is selected for each lane.

4. Upon obtaining the winning bids, the respective demand was applied to achieve the actual cost for each lane.

2.3 Optimization Model

The first and most basic combinatorial auction optimization problem we present is a linear problem consisting of an objective function and three constraints. We let the sets $LANES$ and $BIDS$ represent the set of all lanes/lots in the auction and the set of all received bids respectively. The indexing over these respective lanes is given by $l = 1, 2, ..., m$, for $l \in LANES$, which represents lane l in the auction, and $b = 1, 2, ..., n$; for $b \in BIDS$, which represents bid b out of the set of all bids. Note that there are a total of m lanes and n bids. Note that v_b represents the cost of bid b and we represent the total demand over all lanes with D. In standard form, the program is given by:

$$\begin{aligned} \text{minimize} \sum_{b=1}^{n} x_b v_b \quad & (1) \\ \text{subject to} \sum_{b=1}^{n} x_b = D \quad & (2) \\ \sum_{b=1}^{n} \delta_{lb} x_b = d_l \quad & (3) \\ x_b \in \{0, 1\} & \end{aligned}$$

n: Number of bids
v_b: Cost of bid b
x_b: Decision variable for bid b
δ_{lb}: Variable for lane l
d_l: Demand on lane l
D: Total demand

In the cost minimization problem, we want to minimize the cost given by the summation in Equation (1) where x_b is the decision variable for bid b and v_b is the cost of bid b. The decision variable is either 0 or 1, meaning either the bid is not chosen (0) or chosen (1). Note that in this base model, the bid is only chosen if it is the lowest bid for that particular lane. Thus, the result of this sum equals the sum of the lowest bid on each lane. In Equation (2), we add the constraint that the sum of the decision variables must equal the total demand, meaning we must have exactly 234 decision variables equal to 1, since there are a total of 234 lanes. In Equation (3), the δ matrix is introduced. Consider an example with the δ matrix given by:

$$\delta = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

This example is a 5 by 10 matrix, representing 10 bids placed across 5 lanes. The entries with 1 in row 1 represent the 3 bids that were placed on lane 1. There are zeros in columns 1, 2, and 3 for lanes 2-5 because bids can only be placed on a single lane. The sum of each row l in the delta matrix equals the total number of bids placed on lane l. The sum of each column b equals 1. When δ_{lb} is multiplied by x_b, each d_l in Equation (3) should equal 1.

2.4 Results

The cost minimization model yielded a total cost of \$9,889,386. This cost is down over 1.5 million dollars from last year's total cost of \$10,475,113. Changing the mathematical equation to maximize total cost, the "worst case scenario" yielded a total cost of \$17,045,324, up over 7 million dollars from the minimized total cost.

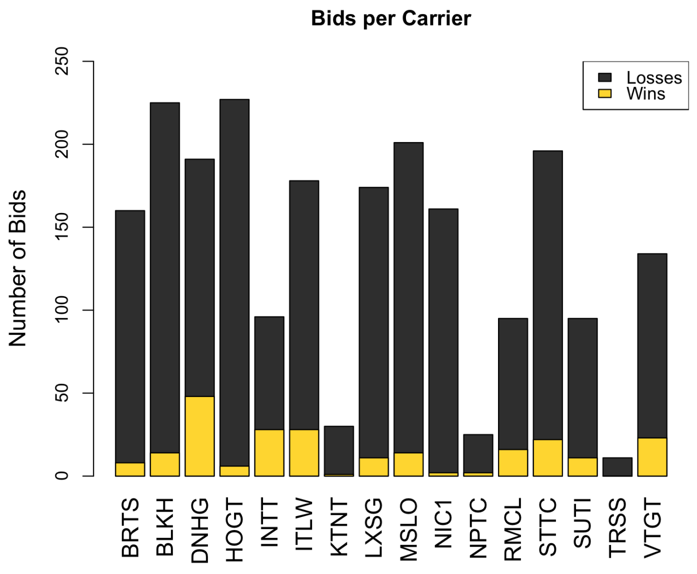


Figure 5: Proportion of bids won by each carrier.

In Figure 5 above, we compare the total number of bids won by each carrier to the total number of bids lost by each carrier. From this, we can see that the carriers who placed more bids did not necessarily win more than those who were more selective in their bidding. Carriers DNHG and INTT won many of the bids they placed, while carriers like NIC1 and HOGT won very few of the bids they placed. We also notice that one carrier, TRSS, placed the least number of bids and won none of them.

In Figure 6, we observe that DNHG receives the greatest percentage of the total cost (27%). However, carrier KTNT is also of interest here. Notice that KTNT has the lowest number of winning bids (a single lane), yet we can see that carrier KTNT accounts for the second-largest percentage of the total cost. The lane which they won, lane 193, has 470 shipments with a cost of $3944.4 per shipment, explaining this large value.

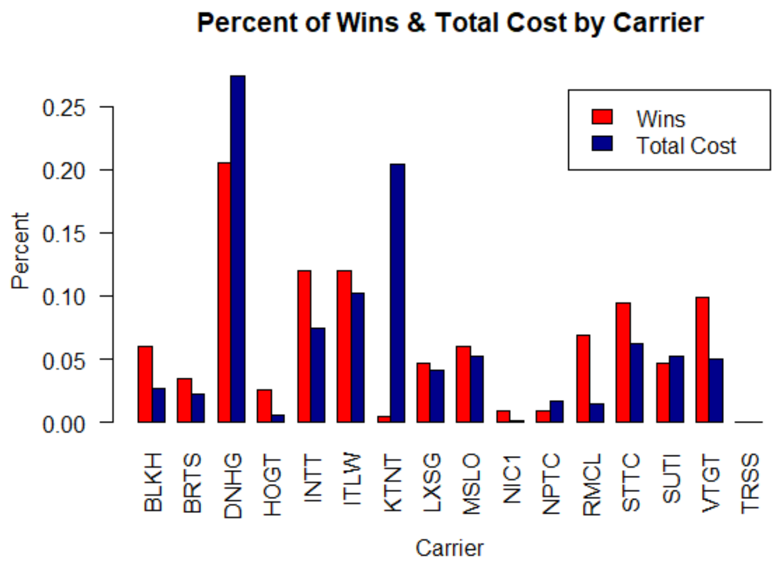


Figure 6: Percent of Wins and Total Cost by Carrier.

The information that can be extracted from our base model results gives an idea of the breakdown of the total cost. We can see which carriers Eastman is relying on the most for distribution and which are contributing the largest amounts to their total expenses. This knowledge could help Eastman in building and maintaining business relations or reducing costs by considering bundled deals with carriers interested in a large number of lanes.

3 Incumbent Constraint Model

3.1 Objective

After constructing and analyzing the base model, we proceeded to apply different constraints to see how the total cost and the chosen carriers changed. The first additional constraint was to force the selection of the incumbent vendor for each lane (if they placed a bid regardless of the cost). The rationale behind this is to reduce business complexities and leverage previous experience with vendors.

3.2 Optimization Model for Incumbents

Again, we present this as a linear problem consisting of an objective function and three constraints. We let the sets $LANES$ and $BIDS$ represent the set of all lanes/lots in the auction and the set of all received bids respectively. The indexing over these respective lanes is given by $l = 1, 2, ..., m$, for $l \in LANES$, which represents lane l in the auction, and $b = 1, 2, ..., n$; for $b \in BIDS$, which represents bid b out of the set of all bids. Note that there are a total of m lanes and n bids. Also, v_b represents the cost of bid b and we represent the total demand over all lanes with D. In standard form, the program is given by:

$$\text{minimize} \sum_{b=1}^{n} x_b v_b \qquad (4)$$

$$\text{subject to} \sum_{b=1}^{n} x_b = D \qquad (5)$$

$$\sum_{b=1}^{n} \gamma_{lb} x_b = d_l \qquad (6)$$

$$x_b \in \{0, 1\}$$

n: Number of bids
v_b: Cost of bid b
x_b: Decision variable for bid b
γ_{lb}: Variable for lane l
d_l: Demand on lane l
D: Total demand

The mathematical model for the incumbent constraint is similar to the cost minimization model. The only difference is that we have replaced the delta matrix with a gamma matrix. Consider an example in which bids were placed on 5 lanes A, B, C, D, and E. If there are 3, 2, 2, 1, and 2, bids on these lanes, respectively, and the incumbent's bid appears first in the bid data, then the resulting gamma matrix would be given by:

$$\gamma = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

3.3 Results

We found that the incumbent vendors placed bids for all 234 lanes, and this gave us a total cost of $11,430,865.00, which is an increase of $1,541,479 from the base model. Interestingly, this represents a 9% (about $1M) increase in cost from last year's total and a 16% increase compared to the base model cost. The increase in cost from last year can be further investigated by Eastman since we are dealing with the same vendors from last year. In Figure 7, we notice some great differences between the base model results and the incumbent constraint model results. Consider vendors INTT and KTNT. When choosing the incumbent constraint model, INTT receives over $4 million, compared to just over $500,000 in the base model. In contrast, KTNT receives almost $2 million when choosing the base model, but receives almost nothing when choosing the incumbent restraint model. This suggests that INTT may be placing less competitive bids this year, while KTNT may be placing more competitive bids this year.

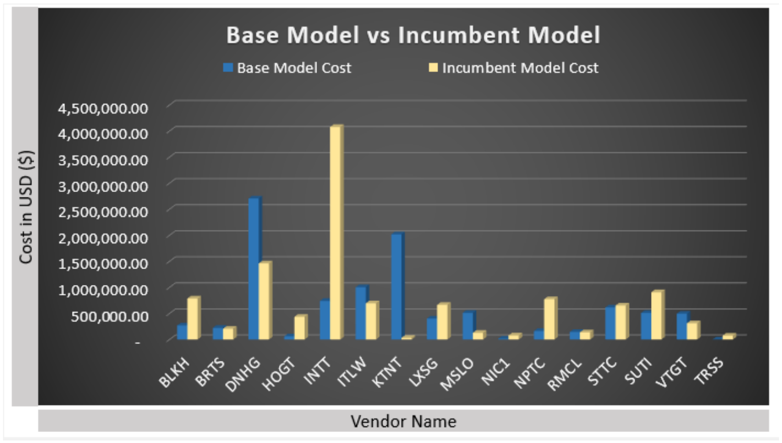


Figure 7: Percent of Wins and Total Cost by Carrier.

4 Cluster Carrier Constraint Model

4.1 Objective

Additional variations in model construction can take on many forms depending on the goal of the distribution of the winning bids. Here, we introduce the Cluster Carrier Constraint Model. In this model, we are considering the subsets of lots by site cluster. Recall there was a Site Cluster column in the Lot Data in Table 1. The individual clusters in our data set include "Benelux", "Germany", "UK," and "Nordics". The Carrier Constraint model places a limit on the number of carriers selected for the winnings bids. In a general sense, our goal is to limit the variation in carriers shipping to the clusters of lots.

Figure 8 shows how the data was separated and then recombined for the cluster carrier constraint model. Specifically, the Bid Data was separated into 4 based on site cluster. Each cluster was worked on separately and the results were later combined.

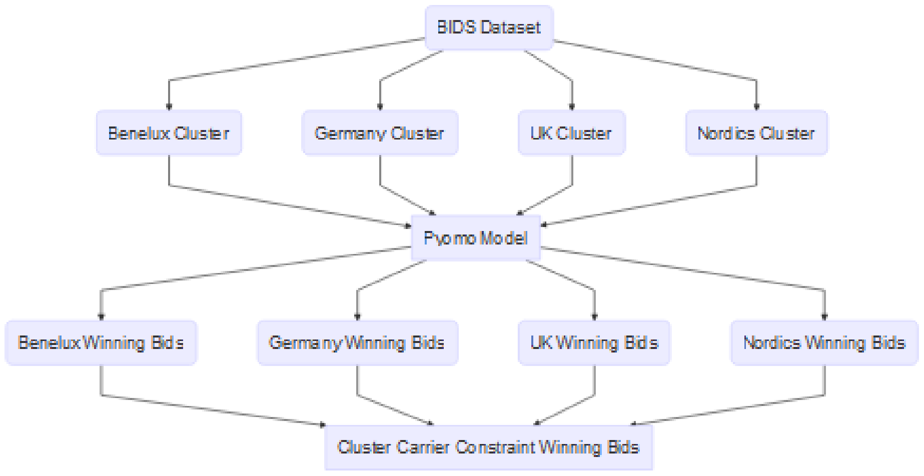


Figure 8: Diagram of how the bids were separated for the cluster constraint.

4.2 Optimization Model for Carrier Constraint

In the carrier constraint model, we set the maximum number of carriers to 5 and the minimum to 1. In order to do this, we introduce a new indexing set $CARRIERS$ which represents the set of all carriers denoted $c = 1, 2, ..., r$ where r is the total number of carriers participating in the tender. Also, two new parameters are introduced, z_c, which tracks which carriers are assigned bids in the final allocation, and

M_c, which denotes the total number of bids placed by carrier c. In summary, for a carrier that places n bids,

$$z_c = \begin{cases} 1 & \sum_{i=1}^{n} x_i \geq 1 \ , \ x_i \in (0,1) \\ 0 & \text{otherwise} \end{cases}$$

Similar to the Cost Minimization and Incumbent Models, we introduce a new incidence matrix for the carriers denoted η where the c^{th} row corresponds to the c^{th} carriers and the b^{th} column corresponds to the the b^{th} bid. Each entry in the η incidence matrix is defined as

$$\eta_{cb} = \begin{cases} 1 & \text{if carrier } c \text{ placed bid } b \\ 0 & \text{otherwise} \end{cases}$$

Also, note that summing over each row of η produces the number of bids placed by each carrier, $M_c = \sum_{b \in BIDS} \eta_{cb} \; \forall c \in CARRIERS$. Combining these new constraints with Cost Minimization Model, we obtain the following linear program:

$$minimize \sum_{b=1}^{n} x_b v_b \tag{7}$$

subject to

$$\sum_{b=1}^{n} x_b = D \tag{8}$$

$$\sum_{b=1}^{n} \delta_{lb} x_b = d_l \; \forall l = 1, ..., m \tag{9}$$

$$\sum_{b=1}^{n} \eta_{cb} x_b - M_c z_c \leq 0 \; \forall c = 1, ..., r \tag{10}$$

$$z_c - \sum_{b=1}^{n} \eta_{cb} x_b \leq 0 \; \forall c = 1, ..., r \tag{11}$$

$$\sum_{c=1}^{r} z_c \geq minCarrier \tag{12}$$

$$\sum_{c=1}^{r} z_c \leq maxCarrier \tag{13}$$

$$x_b \in \{0, 1\}$$

$$z_c \in \{0, 1\}$$

Note that Equations (7) - (9) are the same as the Cost Minimization model which requires that the total demand across all lanes is met by the accepted bids. Constraints (10) and (11) are derived in [2] and are used to determine the carriers in the winning bids. Equations (12) and (13) are predetermined values for the maximum and minimum number of carriers. The decision variables are required to take on values 0 or 1.

4.3 Results

In Figure 9, we can see that this model resulted in several carriers not being chosen for any lanes, and carriers DNHG, INTT, and ILTW being chosen for many. It is also clear from this that the vast majority of the lanes fall within the Benelux cluster, and the fewest belong to the UK cluster.

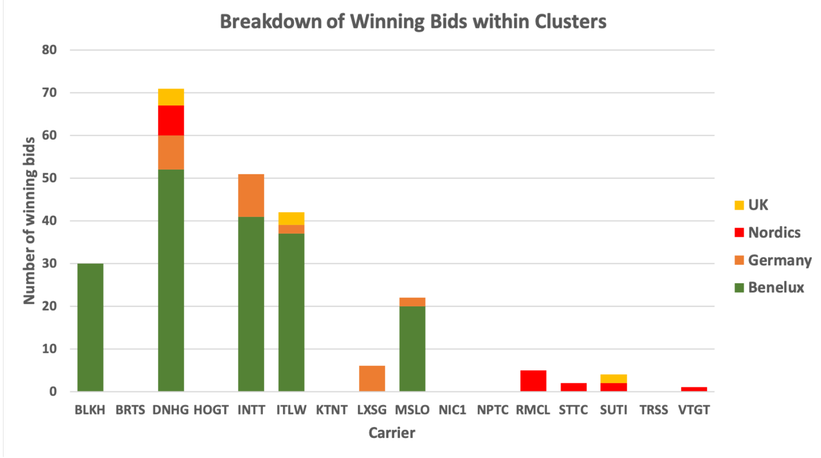


Figure 9: Number of winning bids assigned to each carrier by cluster.

This model resulted in a total cost of $10,159,850.40, an increase of $270,464.40 dollars over the base model. Figure 10 displays the costs per carrier associated with the cluster model. Carriers DNHG and MLSO make up the largest portion of the total cost, and 6 of the 16 carriers were not chosen at all. Additionally, we can see that the bulk of the cost belongs to the Benelux cluster.

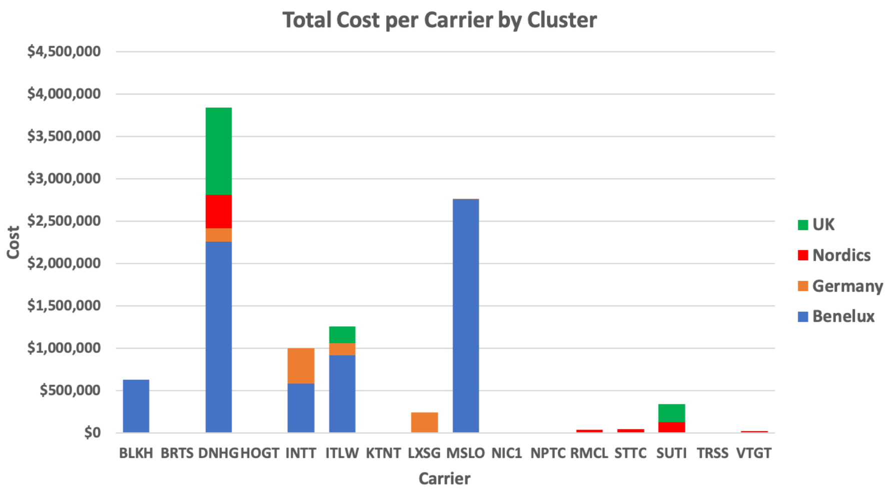


Figure 10: Total cost for each carrier within each cluster.

Figure 11 shows a map of the costs for lanes coming from Benelux using only 5 carriers. It is interesting to see that Benelux, which is a union of the western European states of Belgium, Netherlands, and Luxembourg, has distribution routes to countries all across the globe.

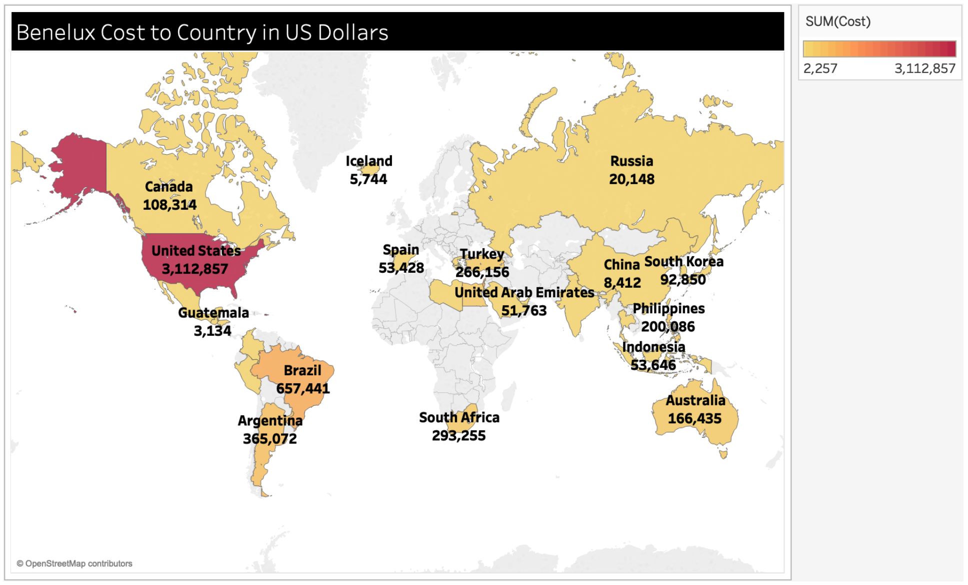


Figure 11: Map of the costs for lanes from Benelux to various countries.

5 Split Constraint Model

Finally, the last constraint was to implement a shipment split where, depending on the splitting formula, the number of shipments on a lane is split between the vendor with the lowest bid and the incumbent vendor. This method is meant to keep costs low while also building and maintaining relationships with carriers. For this model, we simply combined the results from our base and incumbent models and modified the calculations. We used 7 different splitting formula and compared the costs. The totals for each of these is shown in Table 3.

Table 3: Results for various percentage splits.

Split Shipment Calculation	Lowest Bid Cost ($)	Incumbent Vendor Cost ($)	Total Cost ($)
100% Lowest, 0% Incumbent	9,889,386.00	0	9,889,386.00
80% Lowest, 20% Incumbent	8,276,744.40	1,910,959.20	10,187,703.60
60% Lowest, 40% Incumbent	6,640,684.80	3,850,082.40	10,490,767.20
50% Lowest, 50% Incumbent	5,945,796.00	4,672,820.40	10,618,616.40
40% Lowest, 60% Incumbent	4,828,882.80	6,000,601.20	10,829,484.00
20% Lowest, 80% Incumbent	3,192,823.20	7,939,724.40	11,132,547.60
0% Lowest, 100% Incumbent	0	11,430,865.20	11,430,865.20

Here, we see the cost associated to each splitting formula. Notice that 100% Lowest - 0% Incumbent is the Cost Minimization Model while 0% Lowest - 100% Incumbent is the Incumbent Constraint Model. The 80/20 split appears to be the optimal splitting formula in terms of reduced cost as well as uniform distribution of the lanes. As a result, this split was chosen and compared to the base model. From the table, the resulting cost is $10,187,703.60, a cost of $298,317.60 more than the base model.

6 Comparison of Results

Having constructed each of the models, we wanted to compare the distributions of the winning bids and the costs among the carriers. First we look simply at the total cost for each of the four models and compare it to last year's cost and the maximum possible cost.

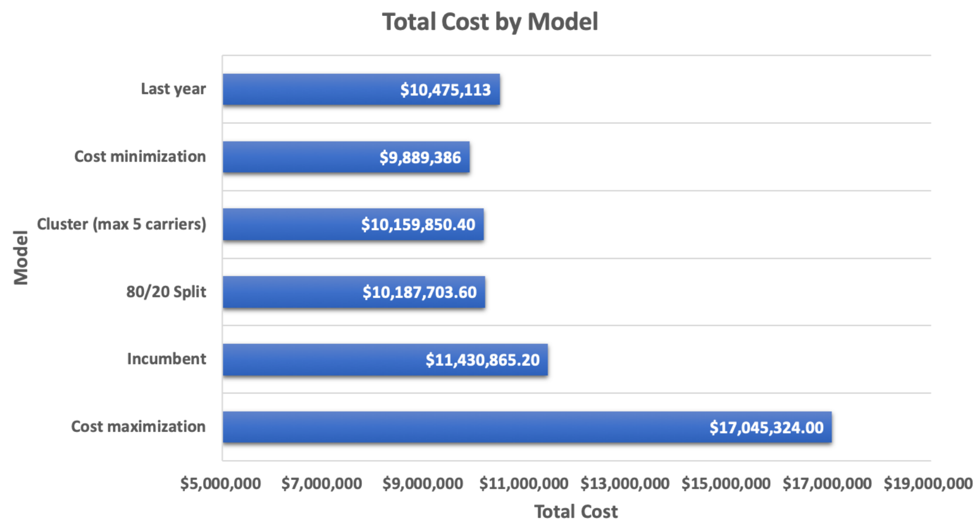


Figure 12: The total cost for each of the four models.

Comparing each model to the base model in Figure 12, we can easily see that while the 80/20 split and cluster constraints increase the cost, these are much smaller increases than the incumbent constraint, which increases the price by approximately $1.5 million. Comparing the incumbent cost to last year's cost, we see there is an increase of nearly $1 million, which could warrant further investigation since the carriers are all the same. Lastly, we can see that none of our models result in costs close to the cost maximization result of $17 million.

Figure 13 shows the number of winning bids for each of the 16 carriers across the four models. There are many observations to be made from this. First, we can see that carriers KTNT, NIC1, and TRSS win the lowest number of lanes in any given model, and in some cases do not win any at all. On the other hand, carriers DNHG, INTT, and ITLW win a large number of lanes regardless of the model chosen. The cluster model, represented by the purple bars, is the most dissimilar from the other models; only 10 of the 16 carriers are chosen using this method. The base model results in 15 being chosen, while both the incumbent and split models result in all 16 carriers receiving at least one lane.

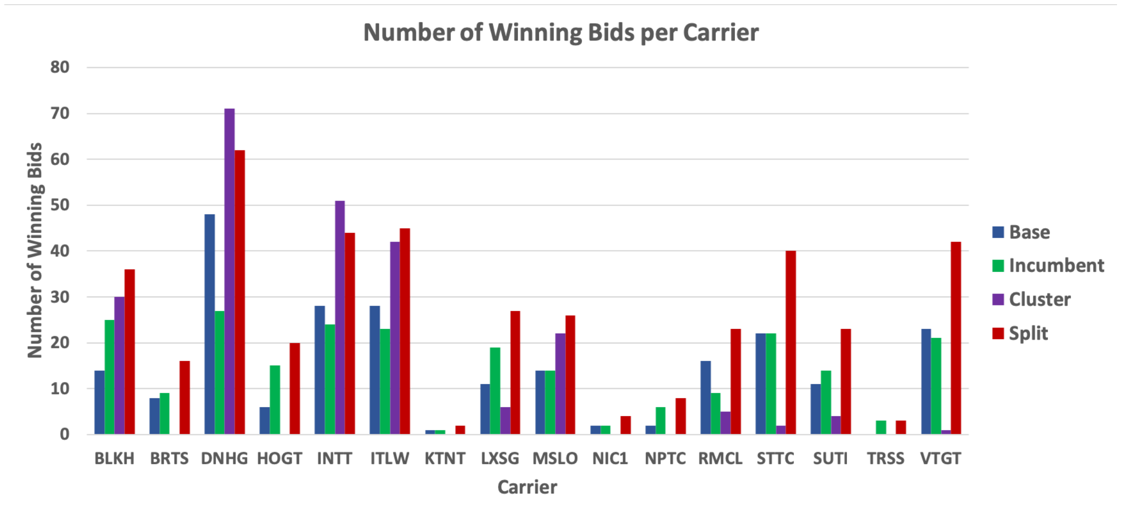


Figure 13: The number of lanes assigned to each carrier, separated by the four models.

Figure 14 gives a comparison of the total amount paid to each of the 16 carriers in each model. While some of the carriers receive similar amounts regardless of the model, others show high variation. INTT, for example, has the highest total cost for any carrier in any model in the incumbent model, a result of them winning the lane with 470 shipments. This lane is also largely responsible for the cost variations for carriers DNHG and MSLO.

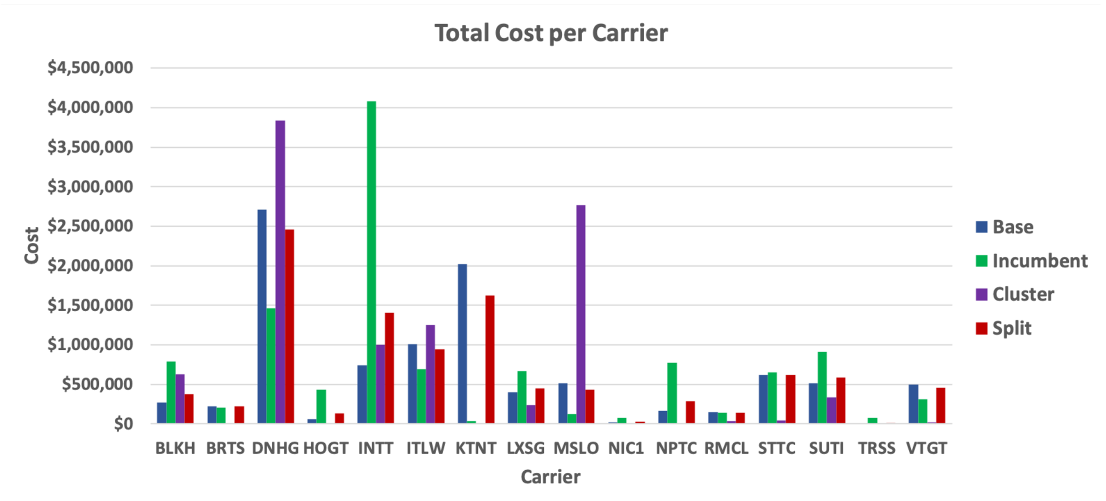
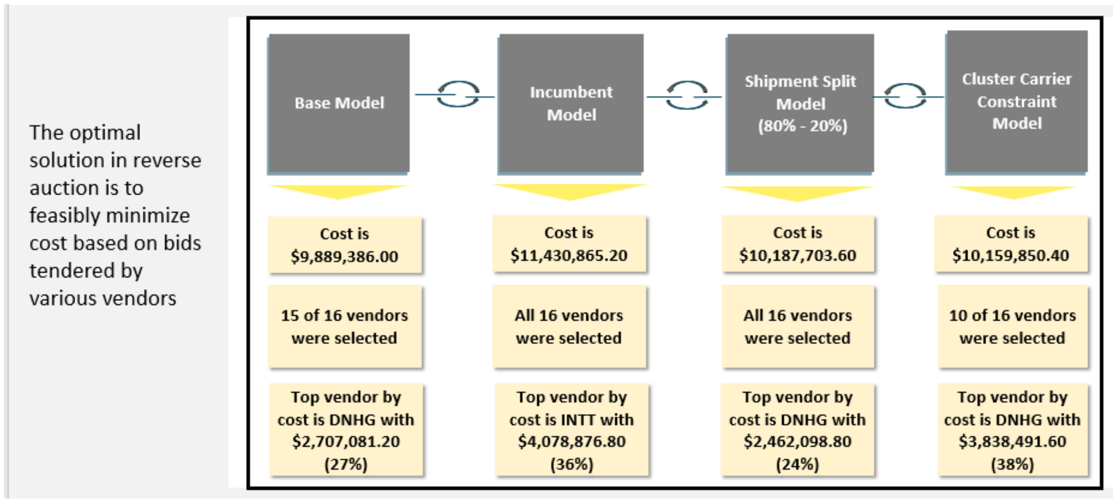


Figure 14: The total amount paid to each carrier, separated by the four models.

7 Conclusion

We have compared the results for four different models - a base model that simply minimizes the cost, an incumbent constraint model that chooses the incumbent carrier for each lane, a cluster carrier constraint model that limits the number of carriers within each site cluster to a maximum of five, and an 80/20 split constraint model that assigns 80% of the shipments for a lane to the lowest bidder and the other 20% to the incumbent. A summary of the results is given in Figure 15.



The optimal solution in reverse auction is to feasibly minimize cost based on bids tendered by various vendors

Base Model	Incumbent Model	Shipment Split Model (80% - 20%)	Cluster Carrier Constraint Model
Cost is $9,889,386.00	Cost is $11,430,865.20	Cost is $10,187,703.60	Cost is $10,159,850.40
15 of 16 vendors were selected	All 16 vendors were selected	All 16 vendors were selected	10 of 16 vendors were selected
Top vendor by cost is DNHG with $2,707,081.20 (27%)	Top vendor by cost is INTT with $4,078,876.80 (36%)	Top vendor by cost is DNHG with $2,462,098.80 (24%)	Top vendor by cost is DNHG with $3,838,491.60 (38%)

Figure 15: The total amount paid to each carrier, separated by the four models.

Our recommendation for which model to use would depend on the goals of the company. If the objective is simply to minimize the total cost, then of course the base model would be the logical choice. However, if the goal were to limit the number of carriers, then the cluster constraint model would make the most sense. If Eastman wished to build and maintain relationships, then the 80/20 split would probably be preferred to the incumbent model, as it still includes all of the possible and incumbent carriers, but at a much lower total cost than the incumbent constraint model.

References

[1] Sheffi, Y. (2004). Combinatorial auctions in the procurement of transportation services. *Interfaces*, 34(4), 245-252.

[2] Kiser, Jennifer, "Developing Optimization Techniques for Logistical Tendering Using Reverse Combinatorial Auctions" (2018). *Electronic Theses and Dissertations*. Paper 3460. https://dc.etsu.edu/etd/3460

A: Cost Minimization Model Pyomo Code

```
#Import of main libraries
from matplotlib import pyplot as plt
import numpy as np
from __future__ import division, print_function
from pandas import read_excel
from pandas import DataFrame
from pandas import ExcelWriter
from pandas import ExcelFile
from pyomo.environ import *
model = ConcreteModel()
BidsDf = read_excel('Logistics Tendering Data for CaseStudy PIC2018.xlsx',
sheet_name='Bids')
LanesDf = read_excel('Logistics Tendering Data for CaseStudy PIC2018.xlsx',
sheet_name='Lots')
deltaDf = DataFrame(np.zeros((len(LanesDf.index), len(BidsDf.index))))
for bid in BidsDf.index:
    deltaDf.at[BidsDf.loc[bid,'Lot#'] - 1, bid] = 1
model.numBids = len(BidsDf.index)
model.numItems = len(LanesDf.index)
model.BIDS = Set(initialize = BidsDf.index.values)
model.LANES = Set(initialize = LanesDf.index.values)
bidValues = dict()
for bid in BidsDf.index:
    bidValues[bid] = BidsDf.loc[bid, 'Cost']
model.bidValue = Param(model.BIDS, initialize = bidValues)
demandValues = dict()
for lane in LanesDf.index:
    demandValues[lane] = 1
model.demand = Param(model.LANES, initialize = demandValues)
delta = dict()
for lane in LanesDf.index:
    for bid in BidsDf.index:
        delta[(lane,bid)] = deltaDf.loc[lane,bid]
model.delta = Param(model.LANES, model.BIDS, initialize=delta)
model.x = Var(model.BIDS, domain = Binary)
def obj_expression(model):
    return sum(model.bidValue[i]*model.x[i] for i in model.BIDS)
model.OBJ = Objective(rule=obj_expression, sense=minimize)
def constraint_rule(model, l):
    return sum(model.delta[l,b]*model.x[b] for b in model.BIDS) <= model.demand[l]
model.xConstraint = Constraint(model.LANES, rule=constraint_rule)
def demand_constraint_rule(model):
    return sum(model.x[b] for b in model.BIDS) >= model.numItems
model.demandConstraint = Constraint(rule=demand_constraint_rule)
```

```
def pyomo_postprocess(options=None, instance=None, results=None):
    model.x.display()
from pyomo.opt import SolverFactory
import pyomo.environ
opt = SolverFactory("glpk")
%timeit results = opt.solve(model)
results = opt.solve(model)
model.solutions.store_to(results)
results.write()
print("\nDisplaying Solution\n" + '-'*60)
pyomo_postprocess(None, model, results)
winningBids = []
index = 0
bidNum = 0
for bids in range(2199):
    if model.x[bids].value > 0:
        winningBids.append(bidNum)
    bidNum += 1
    index += 1
winningBidsDf = bidsDf.iloc[winningBids]
```

B: Incumbent Model Pyomo Code

```
#Import of main libraries
from matplotlib import pyplot as plt
import numpy as np
from __future__ import division, print_function
from pandas import read_excel
from pandas import DataFrame
from pandas import ExcelWriter
from pandas import ExcelFile
from pyomo.environ import *
model = ConcreteModel()
BidsDf = read_excel('Logistics Tendering Data for CaseStudy PIC2018.xlsx',
sheet_name='Bids')
LanesDf = read_excel('Logistics Tendering Data for CaseStudy PIC2018.xlsx',
sheet_name='Lots')
IncumbentVector = np.zeros(len(BidsDf.index))
for i in BidsDf.index:
    if BidsDf.loc[i,'Carrier'] == LanesDf.loc[BidsDf.loc[i,'Lot#']-1,'Incumbent']:
        IncumbentVector[i] = 1
    else:
        IncumbentVector[i] = 0
IncumbentDf = DataFrame(IncumbentVector)
IncumbentDf.columns = ['IncumbentBid']
gammaDf = DataFrame(np.zeros((len(LanesDf.index), len(BidsDf.index))))
for i in BidsDf.index:
    gammaDf.at[BidsDf.loc[i,'Lot#'] - 1, i] = IncumbentDf.loc[i]
model.numBids = len(BidsDf.index)
model.numItems = len(LanesDf.index)
model.BIDS = Set(initialize = BidsDf.index.values)
model.LANES = Set(initialize = LanesDf.index.values)
bidValues = dict()
for bid in BidsDf.index:
    bidValues[bid] = BidsDf.loc[bid, 'Cost']
model.bidValue = Param(model.BIDS, initialize = bidValues)
demandValues = dict()
for lane in LanesDf.index:
    demandValues[lane] = 1
model.demand = Param(model.LANES, initialize = demandValues)
gamma = dict()
for lane in LanesDf.index:
    for bid in BidsDf.index:
        gamma[(lane,bid)] = gammaDf.loc[lane,bid]
model.gamma = Param(model.LANES, model.BIDS, initialize=gamma)
model.x = Var(model.BIDS, domain = Binary)
def obj_expression(model):
    return sum(model.bidValue[i]*model.x[i] for i in model.BIDS)
model.OBJ = Objective(rule=obj_expression, sense=minimize)
def incumbent_constraint_rule(model, l):
    return sum(model.gamma[l,b]*model.x[b] for b in model.BIDS) <= 1
model.xConstraint = Constraint(model.LANES, rule=incumbent_constraint_rule)
def demand_constraint_rule(model):
    return sum(model.x[b] for b in model.BIDS) >= model.numItems
model.demandConstraint = Constraint(rule=demand_constraint_rule)
def pyomo_postprocess(options=None, instance=None, results=None):
    model.x.display()
from pyomo.opt import SolverFactory
import pyomo.environ
opt = SolverFactory("glpk")
results = opt.solve(model)
model.solutions.store_to(results)
results.write()
print("\nDisplaying Solution\n" + '-'*60)
pyomo_postprocess(None, model, results)
winningBids = []
index = 0
bidNum = 0
for bids in range(2199):
    if model.x[bids].value > 0:
        winningBids.append(bidNum)
    bidNum += 1
    index += 1
winningBidsDf = bidsDf.iloc[winningBids]
```

C: Cluster Carrier Constraint Model Pyomo Code

```
from matplotlib import pyplot as plt
import numpy as np
from __future__ import division, print_function
import pandas as pd
from pandas import read_csv
from pandas import read_excel
from pandas import DataFrame
from pandas import ExcelWriter
from pandas import ExcelFile
from pyomo.environ import *
bidsDf = read_csv("cluster Bids.csv")
bidsDf = bidsDf.drop("Unnamed: 0", axis = 1)
lanesDf = read_csv("cluster Lanes.csv")
lanesDf = lanesDf.drop("Unnamed: 0", axis = 1)
CarriersDf = read_csv("Carriers.csv")
CarriersDf = CarriersDf.drop("Unnamed: 0", axis = 1)
deltaDf = DataFrame(np.zeros((len(lanesDf.index), len(bidsDf.index))))
for bid in bidsDf.index:
    deltaDf.at[bidsDf.loc[bid,'Cluster Lot Index'] - 1, bid] = 1
etaDf = DataFrame(np.zeros((len(CarriersDf.index), len(bidsDf.index))))
for bid in bidsDf.index:
    etaDf.at[bidsDf.loc[bid,'Carrier Index'] - 1, bid] = 1
Mc = pd.Series(np.sum(etaDf, axis = 1))
carriers = Mc.to_dict()
bidValues = dict()
for bid in bidsDf.index:
    bidValues[bid] = bidsDf.loc[bid, 'Cost']
demandValues = dict()
for lane in lanesDf.index:
    demandValues[lane] = 1
delta = dict()
for lane in lanesDf.index:
    for bid in bidsDf.index:
        delta[(lane,bid)] = deltaDf.loc[lane,bid]
eta = dict()
for c in CarriersDf.index:
    for bid in bidsDf.index:
        eta[(c,bid)] = etaDf.loc[c,bid]
model = ConcreteModel()
model.numBids = len(bidsDf.index)
model.numItems = len(lanesDf.index)
model.numCarriers = len(CarriersDf.index)
model.BIDS = Set(initialize = bidsDf.index.values)
model.LANES = Set(initialize = lanesDf.index.values)
model.CARRIERS = Set( initialize = CarriersDf.index.values)
model.M = Param(model.CARRIERS, initialize = carriers)
model.bidValue = Param(model.BIDS, initialize = bidValues)
model.demand = Param(model.LANES, initialize = demandValues)
model.delta = Param(model.LANES, model.BIDS, initialize= delta)
model.eta = Param(model.CARRIERS, model.BIDS, initialize= eta, mutable = True)
model.x = Var(model.BIDS, domain = Binary)
model.z = Var(model.CARRIERS, domain = Binary)
model.maxCarriers = 5
model.minCarriers = 1
def obj_expression(model):
    return sum(model.bidValue[i]*model.x[i] for i in model.BIDS)
model.OBJ = Objective(rule=obj_expression, sense=minimize, doc='Objective function
def constraint_rule(model, l):
    return sum(model.delta[l,b]*model.x[b] for b in model.BIDS) <= model.demand[l]
model.xConstraint = Constraint(model.LANES, rule=constraint_rule)
def demand_constraint_rule(model):
    return sum(model.x[b] for b in model.BIDS) >= model.numItems
model.demandConstraint = Constraint(rule=demand_constraint_rule)
def constraint2_rule(model, k):
    return sum(model.eta[k,b]*model.x[b] for b in model.BIDS) -
    model.M[k]*model.z[k]<=0
model.upperBoundConstraint = Constraint(model.CARRIERS, rule=constraint2_rule)
def constraint3_rule(model, k):
    return model.z[k] - sum(model.eta[k,b]*model.x[b] for b in model.BIDS)<=0
model.lowerBoundConstraint = Constraint(model.CARRIERS, rule=constraint3_rule)
def constraint4_rule(model):
    return sum(model.z[i] for i in model.CARRIERS)<=model.maxCarriers
model.zConstraint = Constraint(rule=constraint4_rule)
def constraint5_rule(model):
    return sum(model.z[i] for i in model.CARRIERS)>=model.minCarriers
model.zConstraint2 = Constraint(rule=constraint5_rule)
def pyomo_postprocess(options=None, instance=None, results=None):
    model.x.display()
from pyomo.opt import SolverFactory
import pyomo.environ
opt = SolverFactory("glpk")
results = opt.solve(model)
model.solutions.store_to(results)
winningBids = []
index = 0
bidNum = 0
for bids in range(len(bidsDf.index)):
    if model.x[bids].value > 0:
        winningBids.append(bidNum)
    bidNum += 1
    index += 1
winningBidsDf = bidsDf.iloc[winningBids]
```