ISYE 3133 Semester Project Spring 2021

Zach Ellison, Wyatt Entrekin, Rocco Russo

Formulation

$$min \sum_{a \in A} t_a m_a$$

[1]

$$s.t. \sum_{a \in A_n^+} c_{a,k} - \sum_{a \in A_n^-} c_{a,k} = b_{k,n}$$

$$\forall n \in N, \forall k \in K$$

[2]

$$\sum_{k \in K} (w_k c_{a,k}) + e_a = t_a$$

$$\forall a \in A$$

$$t_a \in \mathbb{Z}^+, c_{a,k} \in 0, 1, e_a \in [0, 1]$$

Choice Parameters	**Definition**
t_a	number of trucks on arc a
$c_{a,k}$	whether commodity k is transported on arc a
e_a	amount of empty space on last truck taking arc a

Set	**Definition**
N	All nodes in network
A	All arcs in network
K	All commodities
A_n^+	All arcs ending at node n
A_n^-	All arcs beginning at node n

Other Variables	**Definition**
m_a	cost of using arc a for one truck
w_k	total weight of commodity k
o_k	source node for commodity k
d_k	sink node for commodity k

Demand Notation	**Definition**
$b_{k,n}$	demand of commodity k at node n such that $b_{k,n} \in -1, 1, 0$

Demand Value	Criteria
$b_{k,n} = -1$	$n = o_k$, or the source node
$b_{k,n} = 1$	$n = t_k$, or the sink node
$b_{k,n} = 0$	otherwise, or at any transit node

Formulation Explanations

Objective Function: The goal of the objective function is minimizing cost of operation, which we have gathered from the the problem description is defined as the sum over all arcs of the cost of using that arc. The cost of using that arc is simply defined as the cost of using a truck on a that arc times the number of trucks using that arc.

Constraints:

1. **Constraint 1** shown above is the conservation of flow constraint, which says that the difference in the flow into the node and out of the node per commodity, over all arcs, must equal the demand of commodity k at node n. We have defined the demand notation and constraints at the bottom, indicating that the only time in which flow is not conserved (demand is 0)are at the source node (demand is -1) and at the sink node (demand is 1)
2. **Constraint 2** shows that the weight of commodity k times whether it is being transported on a certain arc a summed over all commodities, added with the empty space on the last truck taking that arc equals the total number of trucks taking arc a.

Implementation - Graph Parameter - Explanations

For **Commodities** and **Quantities**, we used a dictionary where the keys are commodities and the values are quantities like the hint says because whenever the code is dealing with a certain commodity, it will be very easy to reference the quantity efficiently.

For **Nodes**, we used a dictionary where the key is the node (city name) and the value is a tuple containing the x coordinate first and the y coordinate second. Similarly to the Commodities and Quantities, the coordinates will be very easy to reference.

For **Arcs**, we used a list of tuples where each tuple is (Head Node, Tail Node). When iterating through the set of Arcs, it is very easy to reference either the head or tail and refer to its coordinates in the Nodes dictionary.

For **Sources**, we used a dictionary where the keys are commodities and the values are sources. Like the Commodities and Quantities dictionary, it will be very easy to reference the source of a given commodity.

For **Sinks**, we used a dictionary where the keys are commodities and the values are sinks. This follows the same reasoning as Sources.

For **Distances**, we used a dictionary where the keys are arcs in tuple form (hence making Arcs a list of tuples) and the value is the Euclidean distance calculated from the coordinate tuples.

The code for graph_parameter.py is commented to show where we did the following:

1. Used Pandas to read the CSV files into dataframes.
2. Iterated through the rows of the appropriate dataframes to extract the specific data we needed in each of our data structures.
3. Iterated through our Arcs list to calculate distances.

Implementation - Gurobi Modeling - Explanations

In the function **get_data()**, we use **graph_parameter()** to read in our data, create a few more data structures (a dictionary for demand at each node for each commodity, a dictionary telling us how many

trucks are on each arc, and a list of commodites). We also, set up the demand at each source and sink node to be ready for the model to run.

In the function **build_model()**, we created the model, we created our decision variables, then added our constraints, and lastly set our objective function. To see more specific explanations, **ModelA.py** is well commented and explains which specific lines create specific parts of our Mixed-Integer Program formulation.

In **optimize_model()**, we optimize the MIP and output the results.

In the main part of the program, we wrote some code to format the results and optimal solution into a readable format.

Implementation - Solving - Explanations

We found an objective value solution of $431.9. The runtime of the program was 0.019274 seconds. The optimal solution which describes the path of each commodity and the number of trucks on each arc can be found in **ModelAOutput.txt**.

Analysis - Path A - Explanations

The code for this analysis can be found in **ModelAPathA.py**. The output for the optimal solution can be found in **ModelAPathAOutput.txt**.

After changing the path-making decision variable to be continuous instead of binary, while the number of trucks remained continuous, we found that the objective function value did not change. The runtime was similarly unaffected with both versions of the program running in around 0.01 - 0.09 seconds. The optimal solution remains the same as each commodity takes the cheapest possible path from source to sink. The paths and the number of trucks on each arc do not change from the previous solution.

This makes sense as with a fractional number of trucks there is no incentive to deviate from the optimal path to save money by trying to fully load a truck. As we will further explore in Analysis Path B, the amount of a commodity transported on a path is equal to the number of trucks and as such the path decision making will be identical as there is no incentive to send anything down anything but the most efficient path because with fractional trucks, we are essentially sending the commodity and there is no limit on weight or anything of the sort. Ultimately there is no concern for sending partially empty trucks as instead we are simply sending partial trucks.

Analysis - Path B - Explanations

The number of trucks on each arc is equal to the weight of the commodity being transported on the arc. As such we can rewrite the objective function as $min \sum_{aA} \sum_{kK} m_a w_k c_{a,k}$. This is the minimization of the summation of the cost of sending a truck or the cost of sending a weight of 1 on the arc times the weight of commodity k times the binary variable determining whether k is sent on arc a. Thus our updated formulation is as follows:

$$min \sum_{aA} \sum_{kK} m_a w_k c_{a,k}$$

[1]

$$s.t. \sum_{a \in A_n^+} c_{a,k} - \sum_{a \in A_n^-} c_{a,k} = b_{k,n}$$

$$\forall n \in N, \forall k \in K$$

[2]

$$\sum_{k \in K} (w_k c_{a,k}) = t_a$$

$$\forall a \in A$$

$$t_a \in \mathbb{Z}^+, c_{a,k} \in 0, 1, e_a \in [0, 1]$$

Choice Parameters	Definition
t_a	number of trucks on arc a
$c_{a,k}$	whether commodity k is transported on arc a

Set	Definition
N	All nodes in network
A	All arcs in network
K	All commodities
A_n^+	All arcs ending at node n
A_n^-	All arcs beginning at node n

Other Variables	Definition
m_a	cost of using arc a for one truck
w_k	total weight of commodity k
o_k	source node for commodity k
d_k	sink node for commodity k

Demand Notation	Definition
$b_{k,n}$	demand of commodity k at node n such that $b_{k,n} \in -1, 1, 0$

Demand Value	Criteria
$b_{k,n} = -1$	$n = o_k$, or the source node
$b_{k,n} = 1$	$n = t_k$, or the sink node
$b_{k,n} = 0$	otherwise, or at any transit node

This updated formulation is fairly similar to the formulation for the transportation networks with a objective function defined by cost of transportation times the amount transported with the constraints focusing on satisfying supply and demand. Similarly this model only has one decision variable the decision to transport commodity k on arc a.

Analysis - Truck - Explanations

The code for this analysis can be found in **TruckAnalysis.py**. The output for the optimal solution can be found in **TruckAnalysisOutput.txt**.

The objective value of the solution is now 547.07 which is much larger than the previous objective value. The runtime is also significantly longer taking around 30 seconds instead of only taking less than a second. The increase in runtime and the worsened optimization of the solution is unsurprising as we are now using integer programming instead of linear programming as a result the computer is having to do the simplex method many more times than it did before we had integer trucks. The solution is unsurprisingly less efficient as we now have empty spaces on trucks and additional restrictions that take us farther from the non integer original solution. These empty spaces mean we have to use more trucks which cost more money and the computer also has to calculate far more routes. Many commodities used the same route over both programs

however some changed their route by joining routes where other commodities have extra space on their trucks allowing these commodities to travel more efficiently as now instead of only taking 0.1 of a truck they need a full truck whose wasted space would be wasted money. Unlike before the number of trucks on each arc does not simply match the weight shipped on each arc and instead is the lowest integer value greater than the weight shipped on the arc.

Extra Credit