**SCM 518 – Final Project**

**Team D11: Aditya SV, Sijia Wu, Dongyeon Kim, Rohit Chaudhari**

**Truck Assignment and Scheduling**

**Introduction**

A company which is in the business of delivering ready-mix concrete to its customers is often faced with the task of optimizing its operations. Based on our discussion with the contacts in the company, some of the optimization challenges that this company wants to solve are:

1. Anticipating future orders and optimizing resources to fulfil those orders
2. Whether to accept or reject future orders based on existing capacity
3. Preparing the daily roster of truck drivers and their arrival times to ensure their well-being and longevity
4. Generate best possible schedule to deliver concrete to customer locations
5. Continually update these recommendations based on the changes that occur

**Problem Statement**

The company wants to tackle these issues in isolation to begin with and later merge them in to one big optimization problem. They are interested to know if we can use our skills to recommend the best possible schedule for a given day i.e., they want to know the number of trucks from each plant that should be assigned to each customer.

**Details**

Specifically, in a certain geography the company has 4 plants from where the trucks can be loaded with concrete and delivered to customer location. The company has close to 100 clients in total in that geography which it needs to services.

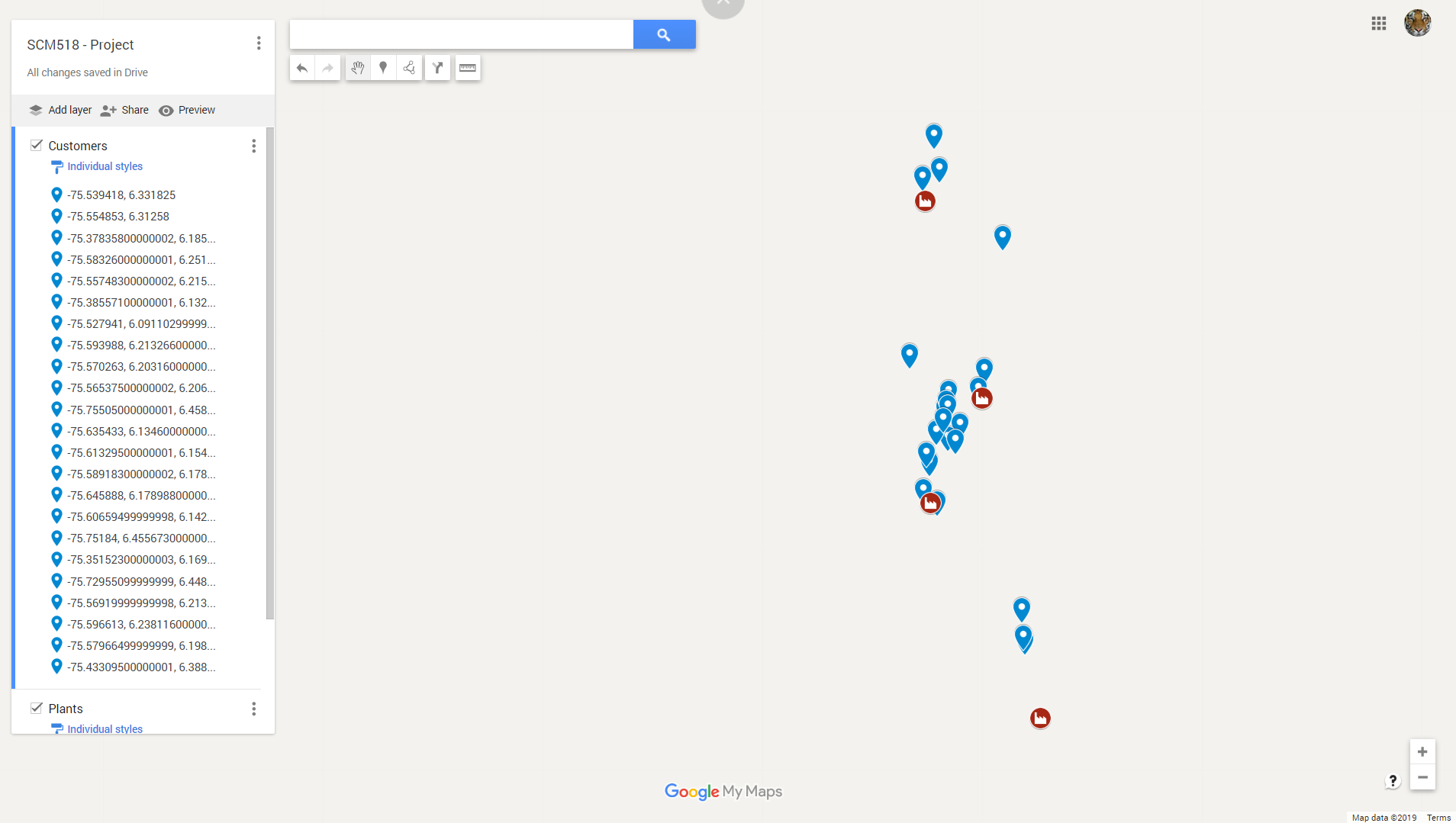
1. Total volume of trucks available on a given day is constant. The company leases trucks based on a demand model and hence the total trucks available may vary from day to day. The total trucks could also vary based on the supply of trucks
2. Each plant has a constant number of trucks available to it every day which should leave at the beginning of the day and return at the end of the day
3. The plants have a fixed concrete loading schedule, for example, every 15 mins, every 1 hour etc. which could be varied on a weekly or monthly basis
4. The plants can only load one truck at a time with concrete
5. On a given day there could be demand from up to 100 customers (i.e. all customers) and each customer could demand more than one load of truck. This is called as ‘Deliveries’.
6. Most importantly, the duration between loading a truck at the plant and pouring the concrete at the customer location should not be more than 90 minutes. This is because waiting for too long can harden the concrete and could potentially lend it useless
7. The deliveries should reach the customers before the requested time. If a delivery reaches after the requested time the order is deemed canceled and loss is incurred to the company. However, if a delivery reaches too early, it is not very efficient since the truck must idle till the delivery time.
8. In the middle of the day, the trucks return to any of the plants to reload the concrete

***The objective is to identify the optimum schedule for the trucks i.e. how should a truck travel between the plants and customers and return back to its base so as to minimize the overall cost associated with the travel and at the same time meet the customer demand on time on a given day.***

**Data**

The following data is gathered from the company:

1. Geographic location of the plants and customers. The latitude and longitude coordinates are provided after adding an offset value to anonymize the locations. Below is snapshot of the relative position of 4 plants and a sample of customers (out of 100).

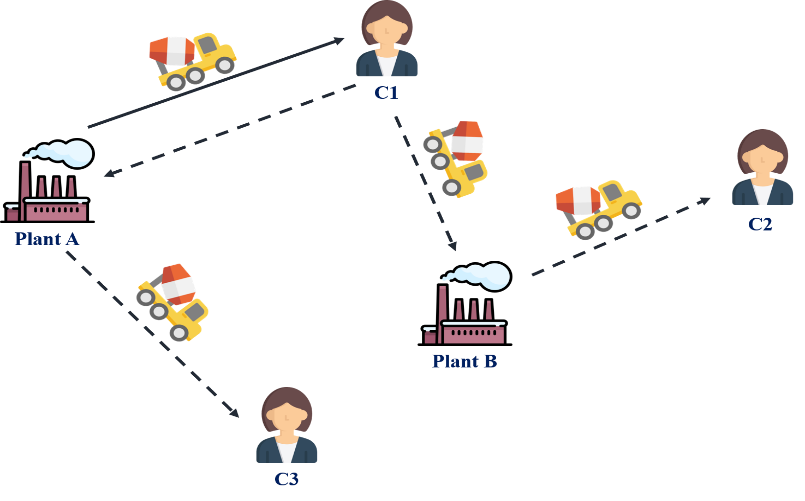


1. Start Time and End Time of each Plant for loading concrete
2. Loading interval at each plant. At this point we assume loading interval to be same for all plants. We can, however, easily incorporate different loading times
3. Starting truck capacity of each plant
4. Delivery requirements on a given day at each customer location. This includes:
   1. Arrival Time
   2. Concrete Pouring Duration
   3. Revenue generated by the delivery

All data is provided in the form JSON files.

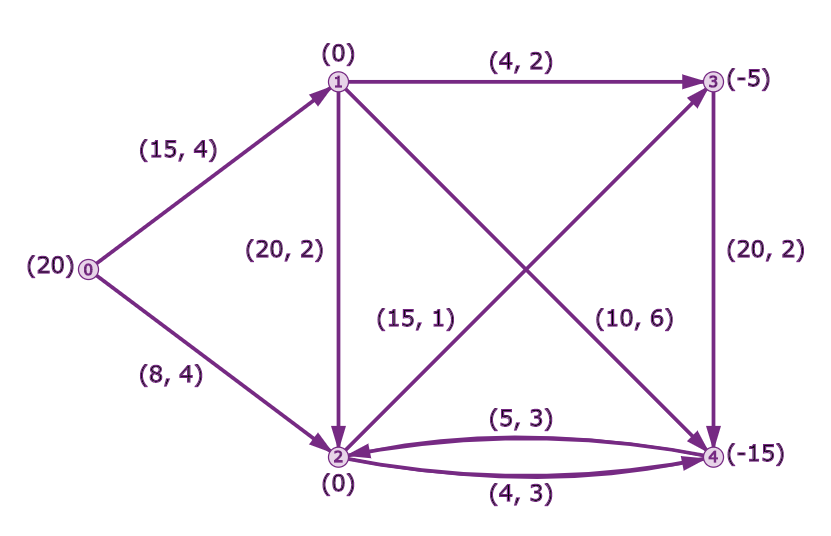
**Approach**

The objective of this problem is to recommend the optimum sequence of actions that a truck needs to take during the day. A truck can leave its base plant and deliver concrete to the nearest customer. But from then on, a truck is faced with multiple choices like whether to go back Plant A or Plant B and which of the many customers to service next.



If we massively simplify this problem, we can recognize that this fits well into a classic network optimization problem called Minimum Cost Flow Problem. As per Wikipedia, “The minimum-cost flow problem (MCFP) is an optimization and decision problem to find the cheapest possible way of sending a certain amount of flow through a flow network. A typical application of this problem involves finding the best delivery route from a factory to a warehouse where the road network has some capacity and cost associated.”

The figure below represents a simple network where the circle are nodes and the lines are the arcs. The number near the nodes represent the supply available or demand. The numbers in parenthesis on the arc represent the capacity and unit cost respectively. In other words, the capacity is the maximum flow of material that the arc can accept, and the unit cost is the cost required to ship a unit of the flow.



In the simplest form, our problem is to find the best delivery route from the plants to the customers by minimizing the cost associated with the travel and meeting the capacity constraints. We have few additional complexities over the simple minimum cost flow model:

1. Deliveries need to be met on time
2. Trucks can revisit plants to reload concrete and continue their journey

After researching about ways to incorporate these additional complexities into the network flow model we are inspired by the ideas presented in an INFORMS journal paper (cited in the references). The idea is to represent the flow, i.e. all possible truck movement ***through time***, between the plants and the customers as a network graph along with quantifying the capacity, unit cost and supply at each node.

Once we can generate this network, we can then formally define the graph and utilize the available tools to solve the problem.

**Generating the Network**

In this section we will detail out all the intricacies of generating a network that captures the movement of trucks on a given day. Before doing that lets define the following terms:

Capacity: the maximum permissible flow (i.e. trucks) on a given arc

Unit Cost: Cost incurred to ship one units of flow (i.e. one truck) along the arc. We will use time as a surrogate for cost

***Super Source and Super Sink Nodes***

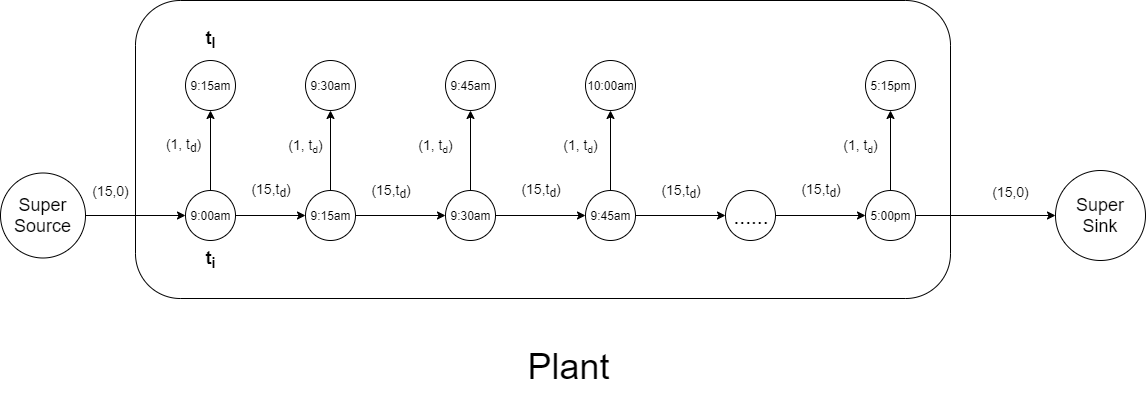
Let’s define two super nodes called super source and super sink. A super source node holds all the trucks at the beginning of the day. Trucks flow from super source to plants in quantities limited by the respective plant capacity. Similarly, a super sink node should collect all the trucks at the end of the day. The figure below represents this idea. Note that super source and super sink nodes are hypothetical and do not exist. This is just for modeling convenience.



Since, these nodes are hypothetical the units cost on these arcs is 0.

***Plant Nodes by Time***

We will represent each plant through time by a series of nodes. In other words, a plant which is represented as node1 at time will transition to node2 (Idle Node) or node3 (Load Node) at time based on whether it loaded the truck with concrete or remained idle, where is the time taken to load the truck.



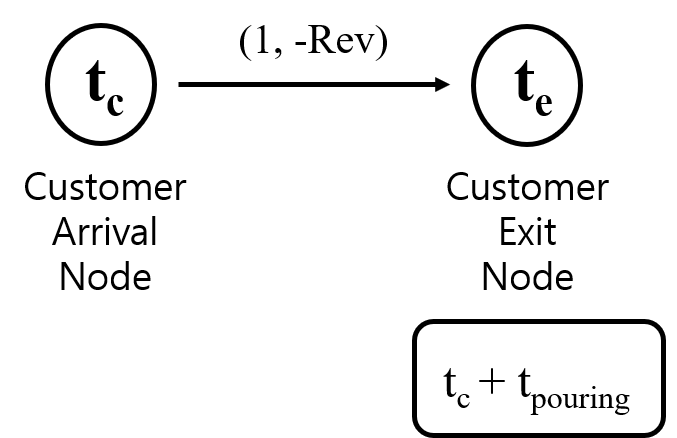


The unit cost in this case will be in both the cases. Since, we can only load one truck at a time the capacity for loading will be 1 on the load arc. However, since any number of trucks can stay idle at the same time, the capacity on the idle arc can be a large number or maximum trucks for that plant.

The trucks that idle till the last time node should flow to the super sink node at the end of the day.

***Customer Deliveries by Time***

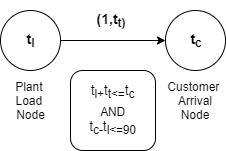
In a similar way we will represent customers with two nodes, one before the delivery called Customer Arrival Node and one after the delivery called Customer Exit Node. Pouring concrete is the profit generating stage for the company, so we will represent that with negative cost. Specifically, we will consider the cost as negative of the pouring time (-Rev). The truck reaches the customer at time and leave the customer at time where is the time required to pour the concrete.



All the customer deliveries are represented by two such nodes which indicates the total customer demand on any given day. The capacity is 1 since only one truck can be poured at a time.

***Load Node to Customer Arrival Node***

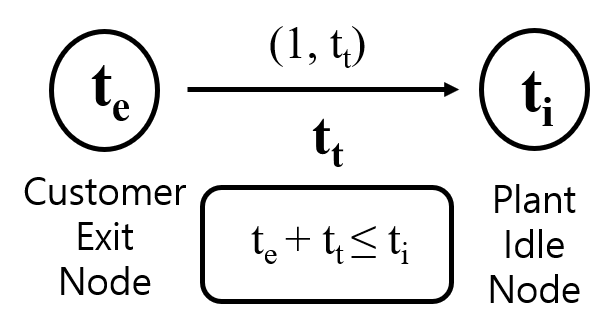
A truck, once loaded, can travel to any customer if the arrival time (current time plus the travel time) falls below the customer’s requested time. Moreover, the duration between loading and pouring should not exceed 90 minutes. So, we create arcs only when both the conditions are met. This can be represented as below:



The unit cost associated will be the travel time from the plant to the customer location. The capacity is 1 since only one loaded truck can travel to the customer location.

***Customer Exit Node to Plant Node***

Once a truck pours the concrete at the customer location, it can return to any plant to reload itself before fulfilling the delivery of the next customer. But the trucks can only return to plants that are still operating (or available to load concrete), i.e. if customer exit time plus the travel time is less than the plant operating time or return to their base if none are operating. This can be represented as below:



The unit cost associated will be the travel time from the customer location to the plant. The capacity is 1 since only one loaded truck can travel to the customer location.

We used python’s network library to construct this graph by bring together these relationships. Such graphs can be constructed very quickly at the beginning of every day based on the customer demand and truck availability. Once the graph is created, we can use it to extract the necessary inputs required to solve the minimum cost flow model.

**Big Picture**

The composite network after bringing all the nodes together looks something like below”

A close up of a map

Description automatically generated

**Minimum Cost Flow Model Set Up**

**Base Model:**

***Parameters:***

: Capacity of the arc between node and node ;

:  Unit cost to transport from node to node ;

: Supplies for node ;

: Number of trucks assigned to Plant k

***Decision:***

:  Amount to flow from node to node ;

***Objective:***

Minimize cost

Min

***Constraints:***

1. ;( Amount to flow should less than the capacity)
2. (net flow in transit nodes should be 0)
3. ; = Super source)
4. ( = Super sink)
5. = Super source and k = Start Node of Plant k)
6. = Super sink and k = End Node of Plant k)

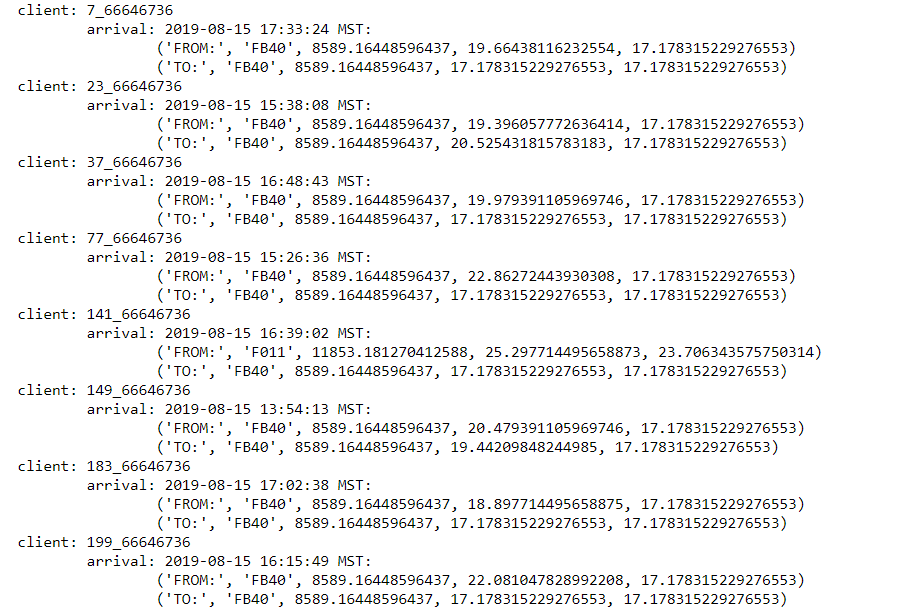
**Model Solving and Results**

We used Google Optimization Tools and Python to solve this model. The output of the optimization was the capacity to flow on each edge and the cost associated with it. We again used python to generate the following two tables which can be easy to interpret and are actionable by the company.

**Table 1:**

The chart below recommends that to serve the order for customer 66646736 at 17:33:24 MST on 2019-08-15, the truck must start from plant FB40 and return to plant FB40.

Similarly, to serve the order for customer 66646736 at 16:39:24 MST on 2019-08-15, the truck must start from plant FB011 and return to plant FB40.

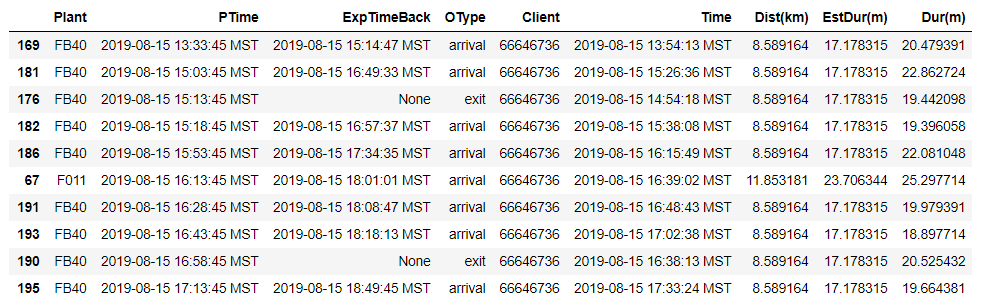




**Table 2:**

For the above two order, the table below shows the start time at the plants to be able to reach the client on time.

For example, the serve order number 67, the truck should leave the plant at 16:13:45 MST and to serve order number 195 the truck the truck should leave the plant at 17:13:45 MST.

****



**Further Improvement**

Utilize Google Maps information to get more recent traffic updates and travel times

**References**:

[1] The Dance of the Thirty-Ton Trucks: Dispatching and Scheduling in a Dynamic Environment. Martin Durbin, Karla Hoffman, Informs Journal