Minimizing Shipping Cost and Redistribution for Recoffery

Eury Sohn
Yonsei University
Creative Technology Management
Seoul, South Korea
ylee12@yonsei.ac.kr

***Abstract*—Recoffery, the start up, manufactures sustainable bioplastic solution by upcycling coffee ground wastes. As the business scales up to venture into their first stage, it is pertinent to have optimized routes for transportations to minimize the costs. This problem will be solved via linear programming.**

I. SPAGHETTI

The impact of spatial networks like streets on human experience cannot be overstated. All of our daily activities take place along or near roads, bike paths, and subway systems, to name a few. As a result, when performing spatial analysis, considering network space rather than Euclidean space allows for a more precise representation of daily human action and movement patterns in many cases. People, for example, do not typically drive in a straight line from their home to work, but rather travel along paths within networks. To this end, spaghetti (spatial graphs: networks, topology, & inference), a sub-module\sembedded in the wider PySAL ecosystem, was developed to address network-centric research questions with a strong focus on spatial analysis (Gaboardi et al., 2018; Rey et al., 2015; Rey & Anselin, 2007).

First, network objects can be created and analyzed from collections of line data using various methods such as reading in a shapefile or passing in a geopandas. Second, spaghetti provides computational tools to aid in the statistical analysis of so-called network-based events across a wide range of previously loaded networks. Network-based events, also known as near-network observations, are events that occur along spatial networks in our daily lives, such as the locations of trees along footpaths, biking accidents along roads, or coffee shop locations along streets. Like spaghetti. Spaghetti, network objects. Shapefiles and geopandas can be used to generate PointPattern objects. Libpy sal.cg or GeoDataFrame objects Pointobjects. Near-network observations can then be snapped to nearest network segments, allowing observation distance matrices to be calculated.

Third, these observation distance matrices can be used within spaghetti to perform clustering analysis or as input for other network-centric problems (e.g., optimal routing), or they can be used within the larger PySAL ecosystem to perform exploratory spatial analysis with esda. Finally, the network elements of spaghetti (vertices and arcs) can be extracted as geopandas. GeoDataFrame objects are used for visualization and are integrated into additional spatial statistical analysis within PySAL (e.g., esda).

II. PROBLEM STATEMENT

There are 8 cafes in Seocho-gu in Gangnam and total of 200 straws for the distribution at the cafes. Different cafes have requested different amount of straws to match their demands and some have requested to refund their straws due to overstocking.

III. METHODOLOGY

This is a "Transportation Problem" where distribution and redistribution of products are required to match the supply and demand of each location, while minimizing the costs. Supply(*n)* and demand(*m*) are represented as unit weights of decision variables at each location The costs are stored in an *n x m* cost matrix.

A. Formulations based on Daskin (2013)

$$\text{Minimize} \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{1}$$

$$\text{Subject To} \quad \sum_{j \in J} x_{ij} \leq S_i \quad \forall i \in I; \tag{2}$$

$$\sum_{i \in I} x_{ij} \geq D_j \quad \forall j \in J; \tag{3}$$

$$x_{ij} \geq 0 \qquad \forall i \in I \quad \forall j \in j. \tag{4}$$

Where i = each potential origin mode

I = the complete set of potential origin modes

j = each potential destination mode

J = the complete set of potential destination nodes

x_{ij} = amount to be shipped from i ∈ I to j ∈ J

c_{ij} = per unit shipping costs between all I, j pairs

S_i = node i supply for i ∈ I

D_j = node j supply for j ∈ J

B. Code

1. Introduction to the Transportation Problem
2. Declaration of Solution class and model parameters
3. Solving transportation problem and the optimal shipment plan

C. Variables to be set

a. Nodes

b. Alpha tag for decisions

c. Shipping costs in matrix

```
# all nodes to be visited
self.supply_nodes, self.demand_nodes = supply_nodes, demand_nodes
# shipping costs (distance matrix) and amounts
self.cij, self.si, self.dj = cij, si.values, dj.values
self.ensure_float()
# alpha tag for decision variables
self.xij_tag = xij_tag
# alpha tag for supply and demand constraints
self.supply_constr_tag = supply_constr_tag
self.demand_constr_tag = demand_constr_tag

# instantiate a model
self.model = mip.Model(" TransportationProblem", solver_name=solver)
# define row and column indices
self.rows, self.cols = self.si.shape[0], self.dj.shape[0]
self.rrows, self.rcols = range(self.rows), range(self.cols)
# create and set the decision variables
self.shipping_dvs()
# set the objective function
self.objective_func()
# add supply constraints
self.add_supply_constrs()
# add demand constraints
self.add_demand_constrs()
# solve
self.solve(display=display)
# shipping decisions lookup
self.get_decisions(display=display)
```

D. Paramteres

```
Parameters
----------
paths : geopandas.GeoDataFrame
    Shortest-path routes between all ``self.supply_nodes``
    and ``self.demand_nodes``.
id_col : str
    ID column name.
ship : str
    Column name for the amount of good shipped.
    Default is 'ship'.

Returns
-------
shipments : geopandas.GeoDataFrame
    Optimal shipments from ``self.supply_nodes`` to
    ``self.demand_nodes``.
"""
```

E. Functions

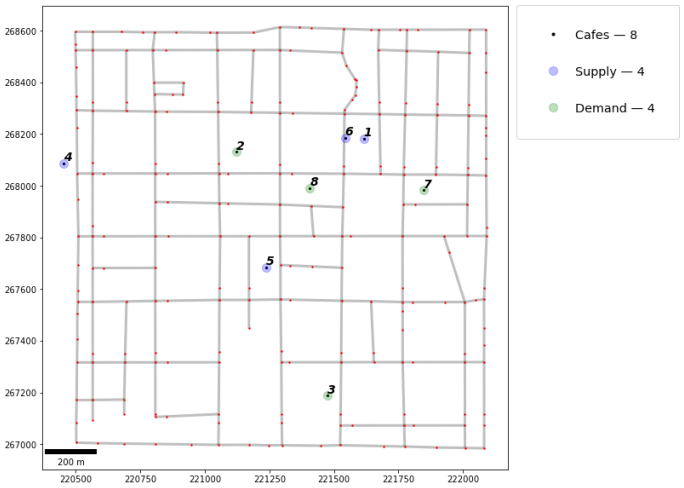
1. Transportation Problem
2. Shipping decision variables
3. Objective function
4. Supply constraints
5. Demand constraints
6. Solve function
7. Shipping decisions

F. Plot Networks in tables

demand	POLYID	geometry
1	2	POINT (221122.271 268131.466)
2	3	POINT (221474.669 267188.462)
6	7	POINT (221847.882 267983.231)
7	8	POINT (221406.839 267990.801)

supply	POLYID	geometry
0	1	POINT (221615.157 268183.063)
3	4	POINT (220453.142 268087.516)
4	5	POINT (221235.835 267685.028)
5	6	POINT (221542.706 268185.028)

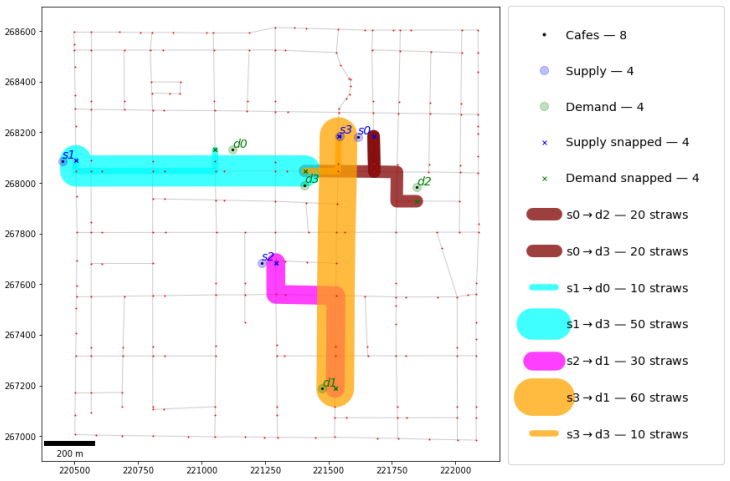
G. Network Plot



To reach the result, it processed through the following steps:

1. Create decision variables for the supply location and mount to supplied
2. Create decision variables for demand locations and amount ot be received
3. Extract network shortest paths
4. Extract shipping paths
5. Plot optimal shipping schedule

IV. Results



The plot above shows the optimal route to minimize shipping costs and redistribute the straws via using linear programming. It demonstrates the starting point and the ending point of the vehicle, with how many straws it should carry during the moving period.