Isaac (Zack) Duitz, Marcus Bluestone

Optimization Methods

6.C57/15.C57, Fall 2024

Optimizing Amazon Last-Mile Routing Logistics

Due: December 9, 2024

Contents

1 Introduction

1.1 The Problem: Amazon Last-Mile Operations

Amazon's e-commerce operations are a cornerstone of its business, characterized by a vast online marketplace offering a diverse range of products, from books to electronics, apparel, and groceries. Its success in this area has stemmed from its dedication to improving customer experience – from the ease of ordering on their website to the speed at which the package arrives at your door.

This paper focuses on improving one specific aspect of Amazon's effectiveness: last-mile operations. This involves where to build stores/facilities to hold the packages before they are shipped to the customer and the routes taken by trucks in order delivery the packages efficiently. This problem is, of course, extremely complex, and a successful solution must balance customer satisfaction, physical limitations, and cost concerns.

We tackle the problem of improving Amazon's current last-mile operations by taking into account concerns about overall cost and customer wait time. We argue that their current facility locations and routes are sub-optimal. More specifically, we show that instead of maintaining a large number of small stores that ship goods to customers, Amazon could merge some of these small stores into larger factories that ship a variety of goods to many customers. In addition, we provide alternative routing logistics that improve efficiency and cost.

1.2 Overview of Approach

First, we parse and interpret our dataset in Section 1.3. Next, we simplify the problem by applying a set of assumptions listed in Section 1.4 to our dataset. Then, we formalize the model mathematically in Section 2.

We first tested our results on a simplified environment (grid world) specified in Section 3.1 and analyzed the runtime. We quickly realized that the problem would be impractical to solve for our large Amazon dataset, so we proposed two strategies for simplifying the problem. We then carry out the experiment in Section 3 and discuss/interpret the results in Section 4. Finally, we propose future work to be done to help generalize, strengthen, and improve the results of this paper in Section 5.

1.3 The Dataset

We use a dataset from Kaggle[1] that provides Amazon's last mile logistics for over 40k deliveries. The dataset contains 43739 rows, corresponding to unique deliveries, and the columns provide the agent age, rating, and the location of the store / drop-off sites.

There are a total of 388 unique stores and 4356 drop-off sites. Moreover, each store is matched with the same drop-off site across all instances. This means that if a drop-off site ordered multiple packages, it comes from the same store each time.

In our model, we save all of the stores and drop-offs and analyze how we can reduce the number of stores and optimize the truck routing in order to minimize both cost and wait-time concerns.

1.3.1 Costs

We set factory costs = 3, truck costs = 1, and hourly wages .1. These choices were somewhat arbitrary and may not align perfectly with reality. They were the product of some experimentation (these values led to quicker solves using Gurobi) and still posses a sensible ordering of cost values (factory costs > truck costs >> wages). Nevertheless, a more robust, concrete approach to this problem would certainly do more research into these numbers and perhaps investigate the quantitative effect that these numbers would have on the overall model. For our limited-scope and limited-capability project, we decided to leave this task as future work. See Section 5 for more information.

[1] https://www.kaggle.com/datasets/sujalsuthar/amazon-delivery-dataset

1.4 Dataset Assumptions

We make a variety of assumptions on the data and real-world logistics in order to simplify the problem.

1. **No Uncertainty:** We assume that there is no uncertainty with regard to any of the data, including costs and demand locations.
2. **Time $\propto$ Distance:** We assume there is a linear relationship between the *spherical distance* between two locations and the time it takes to travel between them. This, of course, is untrue in real-world settings, as a variety of different obstacles (like road locations, traffic controls, and physical blocks) could change the results. However, to estimate true travel-times requires using an API, such as the GoogleMaps API, which would cost over $\$60,000$ to run for all of the stores and drop-offs. (Note that for our simulated environment, explained in 3.1, we use taxicab or L1 distance metrics).
3. **Fixed Factory Cost:** We assume that a large Amazon factory can manage an 'infinite' number of trucks and customers. In other words, the price of maintaining a factory is independent of how many trucks pass through it and how many customers are served by it.
4. **All Drop-off Sites are Equally Important:** In an ideal world, drop-off sites that request a lot of deliveries should be counted as more important when deciding facility locations and truck routes. However, for this problem, we assume that all drop-off sites are equally important and that they each make exactly *one* delivery request.

2 Mathematical Formulation

We formulate our problem as a multi-objective mixed integer linear program.

2.1 Hyperparameters and Data Variables

- We have a set of m candidate factory locations indexed by $i = 1...m$
- We have a set of n demand/drop-off locations indexed by $j = 1...n$
- We have a set of s candidate trucks indexed by $k = 1...s$
- C_f – the cost of building a factory.
- C_t – the cost of buying a truck.
- C_w – the combined hourly cost of truck driver wages and gas costs.
- T^y_{ij} – the time it takes to go from from factory i to demand j.
- $T^x_{j_1 j_2}$ – the time it takes to go from demand j_1 to demand j_2.
- T^z_{ji} – the amount of time it takes to from demand j to factory i.
- α – the weight (between 0 and 1) that we place on the cost component of the objective time.

2.2 Decision Variables

We create binary decision variables indicating which factories are built and which edges are traversed by which trucks. We also have non-negative variables to keep track of the timing of the deliveries.

- $o_i \in \{0, 1\}$ indicates whether factory i is built or not. $(1 \leq i \leq n)$
- $x_{kj_1j_2} \in \{0, 1\}$ indicates whether the kth truck goes from demand j_1 to j_2 or not $(1 \leq j_1, j_2 \leq n,$ $1 \leq k \leq s)$.
- $y_{kij} \in \{0, 1\}$ indicates whether the kth truck goes from factory i to store j $(1 \leq k \leq s, 1 \leq j \leq n;$ $1 \leq i \leq m)$

- $z_{kji} \in \{0, 1\}$ indicates whether the kth truck goes from demand j to factory i ($1 \leq k \leq s, 1 \leq j \leq n$; $1 \leq i \leq m$).
- u_j is a variable representing the position of demand location j in a path of a truck. This is for MTZ constraints to avoid cycles (sub-tours) between demand locations. ($1 \leq j \leq n$)
- $f_k \geq 0$ is the time is takes for truck k to deliver the last package on its route.
- $L \geq 0$ is the latest time it takes for any of the demand locations to receive their package.

2.3 Objective Function

We will have a multi-objective which minimizes over total cost and the latest wait time across all customers.

2.3.1 Objective 1: Cost

The cost component of the objective contains three elements. We formalize this mathematically below.

1. The total cost of building each chosen factory.
2. The total cost of buying each chosen truck.
3. The total cost of paying hourly wages/gas prices. We calculate the total travel time of all vehicles by summing up the corresponding times for each used edge.

$$\min C_f \sum_{i=1}^{m} o_i + C_t \sum_{k=1}^{s} \sum_{i=1}^{m} \sum_{j=1}^{n} y_{kij} + C_w \sum_{k=1}^{s} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} (y_{kij} T^y_{ij} + z_{kji} T^z_{ji}) + \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} x_{kj_1j_2} T^x_{j_1j_2} \right)$$

2.3.2 Objective 2: Latest Wait Time

The justification for minimizing this measure of customer wait time, as opposed to the average or total customer wait-time, is two-fold. First, the cost objective already contains a term that takes into account the total travel time of the trucks. Second, formulating the objective this way encourages the model to find *many* efficient routes that reach multiple customers at once, as opposed to a *small number* of routes that may reach some customers super quickly and others much later. In real-world settings, the former is intuitively more valuable.

The objective function for time will just be to minimize L

$$\min L$$

2.3.3 Multi-Objective Formulation

We use a weight-based approach to formulate the multiple objectives. We define v_1 to be the value of the cost objective and v_2 to be the value of the wait time objective. Our overall objective becomes:

$$\min \alpha \cdot v_1 + (1 - \alpha) \cdot v_2$$

See Section 5 for a longer discussion about this formulation.

2.4 Constraints

2.4.1 Decision Variable Constraints

The decision variables corresponding to building factories or which trucks which edges are all binary. In addition, we have that f_k and L represent times are thus non-negative real numbers. Thus:

$$1 \leq k \leq s, \quad 1 \leq j \leq n; \quad 1 \leq i \leq m :$$

$$o_i, x_{kj_1j_2}, y_{kij}, z_{kji} \in \{0,1\}$$
$$f_k \in \mathbb{R} \geq 0$$
$$L \in \mathbb{R} \geq 0$$

2.4.2 Truck Constraints

We formulate constraints to ensure that used trucks begin and end at existing factories.

1. **Trucks start at most once**. For each truck, the total number of outgoing edges across all factories is at most 1. It is 0 if the truck is unused.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} y_{kij} \leq 1 \quad \forall k \in \{1,\ldots,s\}$$

2. **Trucks must start/end at a factory**: The total number of edges a specific truck can traverse must be less than or equal to the total number of possible edges a truck could take $(n-1)$ multiplied by an indicator whether the truck started at a factory or not. This ensures that truck k can only traverse edges if it first leaves from a factory.

$$\sum_{j_1=1}^{n}\sum_{j_2=1}^{n} x_{kj_1j_2} \leq (n-1) * \sum_{i=1}^{m}\sum_{j=1}^{n} y_{kij} \quad \forall\, k \in \{1,\ldots,s\}$$

3. **Trucks start from *existing* factories**. At most n trucks can leave a single factory:

$$\sum_{k=1}^{s}\sum_{j=1}^{n} y_{kij} \leq n * o_i \quad \forall\, i \in \{1,\ldots,m\}$$

4. **Trucks start and end at the *same* factory**. We set each truck's starting location equal to its ending location. This is a simplification we impose so that factories won't need to worry about their trucks moving locations between days.

$$\sum_{j=1}^{n} z_{kji} = \sum_{j=1}^{n} y_{kij} \quad \forall\, k \in \{1,\ldots,s\} \quad \forall\, i \in \{1,\ldots,m\}$$

2.4.3 Flow Constraints

We formulate the construction flow conditions such that every node (demand location or factory) has exactly 1 truck going in and coming out of it. This will also ensure that all demands will be satisfied.

1. **Total Incoming Flow = 1:** The incoming flow includes flow *from* factories and *from* other demand locations.

$$\sum_{k=1}^{n}(\sum_{i=1}^{m} y_{kij_2} + \sum_{j_1=1}^{n} x_{kj_1j_2}) = 1 \quad \forall\, j_2 \in \{1,\ldots,n\}$$

2. **Flow In = Flow Out for each Truck** If a truck enters a specific node, it must exist that node as well. This implies that the Flow Out = 1 for each node.

$$\sum_{i=1}^{m} y_{kij} + \sum_{j1=1}^{n} x_{k,j_1,j} = \sum_{i=1}^{m} z_{kji} + \sum_{j2=1}^{n} x_{k,j,j_2} \quad \forall\, j_1 \in \{1,\ldots,n\}, \forall k \in \{1...s\}$$

3. **Avoiding Cycles (MTZ Constraint):** We needed constraints to avoid cycles between demand locations which will guarantee that demand locations are on a path containing a factory.[2]

$$1 \leq U \leq N$$

$$u_{j_2} - u_{j_1} \geq 1 - N * (1 - x_{kj_1j_2}) \quad \forall j_1 \in \{1, \ldots, n\} \quad \forall j_2 \in \{1, \ldots, n\} (j_1 \neq j_2) \quad \forall k \in \{1, \ldots, s\}$$

2.4.4 Time Constraints

We defined f_k as the time it takes for truck k to finish its route. This is equal to the sum of the times it took truck k to traverse the edges in its route (excluding the edge from the last demand location to the factory).

$$f_k = \sum_{i=1}^{m} \sum_{j=1}^{n} (y_{kij} * T_{ij}^{y}) + \sum_{j_1=1}^{n} \sum_{j_2=1}^{n} (x_{kj_1j_2} * T_{j_1j_2}^{x}) \quad \forall k \in \{1, \ldots, s\}$$

The latest wait time of the customers is simply the maximum over all f_k:

$$f_k \leq L \quad \forall k \in \{1, \ldots, s\}$$

3 Experiments & Simplifications

3.1 Grid Simulation

We began by running the model on small, simulated environments to gauge its efficiency. We created a $size \times size$ grid, and randomly chose m locations to be candidate factories, n locations to be demand locations, and s trucks to be used. We chose an α value of 0.5, $C_f = 1, C_w = 1, C_t = 3$ (chosen for simplicity), and calculated the T^* values by using Taxicab distance (i.e $|x_1 - x_2| + |y_1 - y_2|$). See Figure 1 for examples results.

However, the solver was extremely slow. For even a small number of trucks and factories, slowly increasing the number of demand locations led to an exponential growth in runtime. See Figure 2 for results.

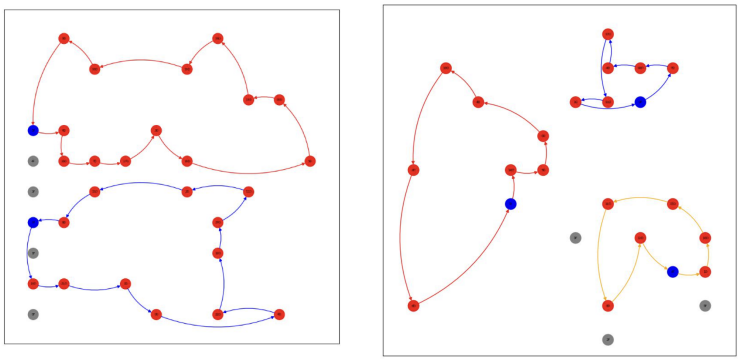

Figure 1: Red nodes are demand locations, blue nodes are chosen factories, and gray nodes are unchosen factories. Each truck route is colored a different color for clarity. **LEFT**: The figure on the left ran with $m = 7$, $n = 33$, $s = 3$. The optimal solution involves picking only 2 factories and using only 2 trucks. **RIGHT:** The figure on the right ran with $m = 6, n = 19, s = 3$. The optimal solution is was to use 3 factories and 3 trucks.

[2] https://how-to.aimms.com/Articles/332/332-Miller-Tucker-Zemlin-formulation.html

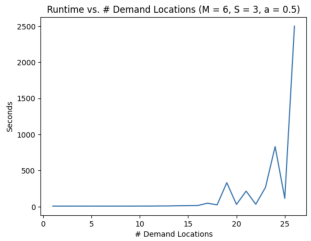

Figure 2: Runtime for the model using parameters specified in 3.1 (and $s = 3$, $m = 7$) as a function of n, the number of demand locations. The simulation was run 3 times and the results were averaged. The abrupt, exponential growth as n increases demonstrates the infeasibility of *fully* running this model for the Amazon dataset.

3.2 Amazon Dataset

We attempted to run the model on the full Amazon dataset ($N = 4356$, $M = 388$, $S = 5$).

We chose parameters $C_f = 3$, $C_t = 1$, $C_w = 0.1$.

In addition, we set $\alpha = 0.5$. This choice balanced the two objectives (minimum latest wait time for customers and overall cost) evenly. A more robust experiment would develop a Pareto front of possible solutions given a range of α values. However, this project was severely limited by compute time and capabilities. See Section 5.

The model included over 1 billion variables (at least $N^2 \cdot S$ demand-demand edges) and Gurobi could not even set up the model after 12 hours of running.

We, therefore, simplified our problem further in 2 ways.

3.2.1 Simplification #1 - Pruning Unreasonable Edges

We inspected the data visually and found that the data from the Amazon dataset was clustered into approximately 24 regions. See the left picture of Figure 4 for visuals. Note that each clusters contains both stores and demand locations.

Realistically, it is unlikely for there to be many interactions between the different regions. We, therefore, decide on a threshold distance value and prune all edges in our model that have a distance greater than this threshold. We experimented with different threshold values to see how many variables our model would have, and saw that decreasing the threshold significantly reduces the number of edges in our model. See Figure 3.

However, we faced two issues with this approach. First, choosing very small threshold values returned an infeasible model. We had no simple way of determining which threshold value would be both feasible and reasonably sized. We attempted to use the maximum distance that any truck traveled in the original dataset but this led to too many variables. Second, even small, feasible threshold values contained impractical numbers of variables ($\approx 10^8$) and Gurobi could not even set up the model after 12 hours of running. We, therefore, required a more efficient solution

3.2.2 Simplification #2 - Clustering

Instead of pruning the edges that were too far, we applied DBSCAN clustering to cluster the different regions in the graph. The results are shown in Figure 4. We then ran our model on each of these 24 regions independently.

On average, each of the regions had approximately $M = 20$ stores and $N = 220$ demand locations. We ran the model with $C_f = 3$, $C_t = 1$, $C_w = 0.1, S = 4, \alpha = 0.5$ for 1 hour and took the best solution that was found (even if it wasn't optimal). See Section 4 for the results.

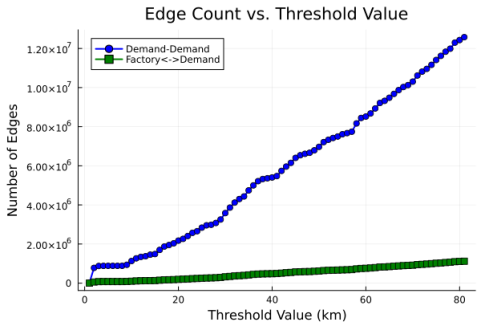

Figure 3: The number of edges in our dataset as a function of the pruning threshold. Notice that the number of edges decreases nearly linearly with threshold value.

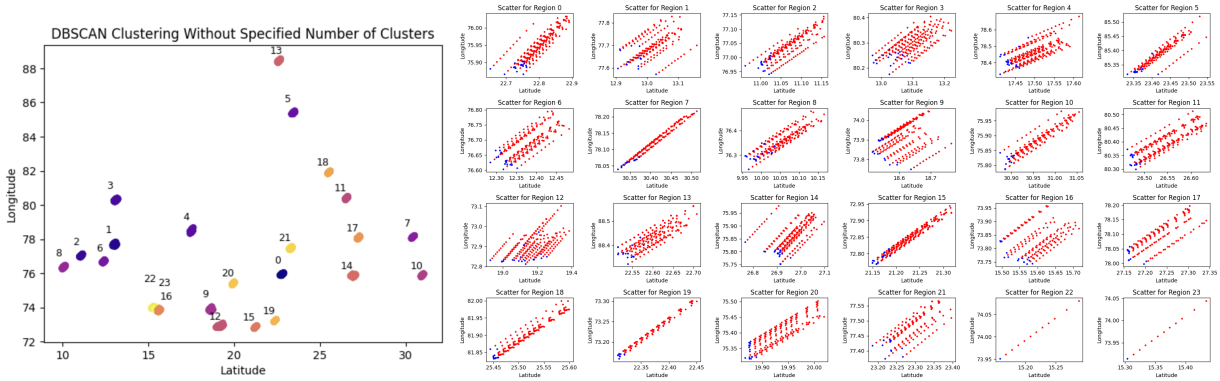

Figure 4: After applying DBSCAN without specifying the number of clusters, we find that there are 24 clusters of data-points (both demand/store sites), where each region consists of demand locations arranged in lines that correspond to streets.

3.2.3 Baseline - Greedy Approach

Because the dataset we used did not explicitly mention the routes taken by the Amazon trucks, we use a greedy routing algorithm as a baseline. However, we do know that every store mentioned in the dataset was used at least once to serve a drop-off location. We thus use the following greedy algorithm:

For each cluster, we look at all of the stores and drop-off spots in that region. For each demand location, we assign the factory that is closest to it. Then, for each factory, we send out one truck to all of its demand locations, where the demand locations are ordered by looking at the next closest unvisited demand site.

4 Results & Implications

We compare the results obtained by the baseline greedy approach and our model's optimal solution. We summed up the total distances traveled and the cost-based metrics across the clusters, and we took the maximum of the latest customer wait time across all the clusters.

We found that the optimal solution involved building only a single factory in each region. In addition, the maximum number of trucks (i.e. 4 trucks) were used in each region. We detail the comparative results below (Table 1) and provide an output solution for one of the clusters (Figure 5).

Our optimized model significantly outperforms the greedy approach. Using $\approx 10\%$ of the number of factories and $\approx 45\%$ of the number of trucks, the optimized model produces a solution that only increased the maximum customer wait time by a factor of ≈ 4 while it reduced the total distance traveled by a factor of ≈ 9.

Metric	Greedy Baseline	Cluster Optimized
# Factories Used	215	24
# Trucks Used	215	96
Total Distance Traveled (km)	42,599	5,753
Max Customer Wait Time (km)	21	94
Final Objective ($\alpha = 0.5$)	597	225
Total Cost	1,173	157

Table 1: Comparison metrics between the greedy baseline and our clustered optimization model rounded to the nearest whole number. Note that wait time is measured in km as mentioned in Section 1.4.

These results indicate that optimizing truck routes and factory placements can allow Amazon to reduce the number of factories it has to operate and save tremendously on overall costs while only increasing customer wait times by a small amount relatively. Although a lot of assumptions were made in order to run this model efficiently, we believe our results are still compelling and telling. Amazon can certainly use a smaller number of factories to provide the same / even better service to its customers.

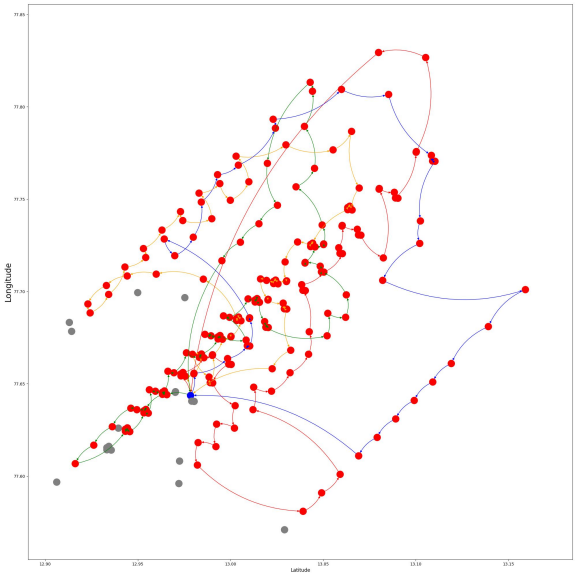

Figure 5: This is the output graph for one of the clusters. Notice that 1 factory and 4 trucks are used.

5 Future Work & Improvements

We recognize the assumptions and simplifications our paper made in order to achieve efficient, accurate results. Here, we propose future work to help improve the robustness, accuracy, and practicality of our results:

1. **Choice of Cost Values:** As noted throughout the paper, our choice of certain hyper-parameters, such as C_f, C_t, C_w, and were not perfectly justified. Although there was some intuition behind our choices, exploring how the results may quantitatively/qualitatively change as a result of shifting the parameters would be a highly relevant task.
2. **Multi-Objective Pareto Front:** Because we are optimizing for both cost and maximum customer wait time, our multi-objective formulation uses an α parameter to control the trade-offs in cost and customer wait time. Ideally, a range of α values should be used and a Pareto Front should be plotted, but because of time and compute limitations, we only tested the implementation for $\alpha = 0.5$

3. **Multi-Objective Goal Programming**: Instead of formulating the multi-objective using a weight-based approach, using a Goal Programming approach (constraining the value of the second objective while optimizing the first) could lead to different or more efficient results.
4. **Robustness:** This paper also made a wide range of assumptions about the fixed nature of the demand locations and hyper-parameter cost values. A stronger model would be robust to perturbations in the data. We were unable to try this due to time and CPU constraints.
5. **Improved Hardware Specs**: Even when clustering, we only ran our model for 1 hour on each region, even if the optimal value wasn't found yet. With a higher number of available threads and more time, we would be able to run our model until the true optimal solution is found for each region.
6. **Using True Times**: With increased compute power and a higher budget, we could use the Google Maps API (instead of spherical distance) to get more accurate values for the time it takes to travel between locations.