Rotman

RSM-8423 – Optimizing Supply Chain Management and Logistics

Winter 2025

HOT DELIVERY

A Case Study in Online Food Delivery Platforms

Source: https://www.appclonescript.com/online-food-delivery/

Introduction

Food delivery platforms such as *Uber Eats*, *DoorDash*, and *SkipTheDishes* have revolutionized the way how people order groceries and interact with restaurants, with an impressive year-on-year growth. For example, by the end of 2021 Uber Eats had already a market share of 29% of all the food delivery services across the United States, surpassing $8 billion in revenues and $30 billion in gross bookings (more statistics here). The second largest company in this space is DoorDash, who reported $4.8 billion in revenues in 2021 but, because of its aggressive restaurant and pricing strategy, controls 45% of the US food delivery market (stats here). (Of note is that Uber has made several attempts to merge with DoorDash in the past.)

While these companies have impressive growth and revenue, surprisingly Uber Eats, DoorDash, and other similar services *have never reported any profits, losing money every year*. This occurs because (a) they are primarily focusing on expanding the business and invest heavily in ads, promotions, and discounts; and (b) very little profit is made per order. The biggest expense delivery companies face is *paying drivers*, who receive a base fare based primarily on the **distance** they traverse. Thus, online food delivery companies invest heavily on data and routing prescriptive models to reduce their delivery costs, bringing to light new challenges in the field.

In this case study, your team was hired by the data analytics group at Uber Eats as consultants to propose a solution to a real-world *strategic* and *operational* problem that they face every day. Specifically, *how to assign drivers to orders based on estimates of their best-possible delivery routes.*

To help you in the analysis, the problem is broken into four parts that incrementally add more of the real-world features of the problem. Distance estimates are based on centroids of Toronto neighborhoods, a common practice where data is readily available. We suggest that you follow the flow provided by Parts I-IV to organize your work and evaluate trade-offs, highlighting the managerial insights you learn throughout the process. All problems should be formulated as mixed-integer linear programs and solved using PuLP (Gurobi is also acceptable if you feel adventurous).

Part I – Single-driver perspective

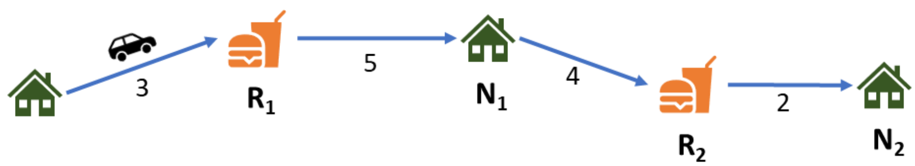
Once a delivery order arrives at the system, Uber Eats assigns it to a driver who picks the food at the restaurant and brings it to the customer. Uber Eats wishes to preemptively assign the driver to multiple pickups, based on the expected time the food will be prepared by the restaurant and the restaurant-customer distance. While the resulting order assignment may not be necessarily shared with the driver at once, it plays a critical role in estimating the best costs required to complete the delivery.

Your objective in Part I is to design a model that produces the optimal delivery route for a single driver, assuming their next few future orders have already been assigned by another system. Your solution must satisfy the following constraints and objective:

- The driver starts at the **Downtown Toronto (Rosedale)** neighborhood.
- The driver can carry multiple orders in their vehicle or transportation modal.
- After finishing all deliveries, the driver parks the car in the neighborhood where the last order has been placed, waiting for future orders.

- Your objective is to **minimize the total distance traversed by the driver**.

For example, the picture below depicts the example of a solution with two orders: one from restaurant R_1 to be delivered in neighborhood N_1, and another from restaurant R_2 to be delivered in neighborhood N_2. Distances are in kilometers. The driver first visits restaurant R_1 and delivers the food to N_1, and then picks up the food at R_2 and delivers to N_2. The total distance traversed by the driver in this solution is 3+5+4+2 = 14 km. Other solutions are possible, such as picking up both orders before delivering them.



Use *part1_ordersB.csv* dataset to evaluate your model. Discuss your solution and describe the route and total distance.

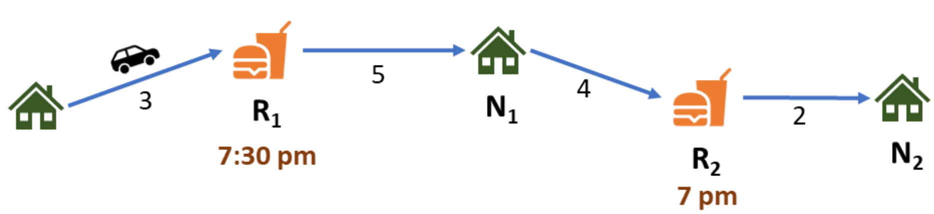
Part II – A matter of time

One of the main challenges in food delivery is *time*. First, the restaurant may require several minutes to prepare the order. Second, the time elapsed between the order being ready for pick up and when the customer receives it cannot be excessively high, which could be detrimental to the quality of the service and the food experience.

Your objective in Part II is to incorporate time into the model you developed in Part I. Specifically, every restaurant now places as estimated time when the food will be ready to pick up. Based on this value, we define the **customer wait time** as the time between when the food is ready to pick up, and when the driver arrives at the customer's address (in this case the neighborhood). Notice that this time does not consider food preparation/packaging because it cannot be optimized by Uber Eats.

Moreover, assume that the driver spends, on average, five minutes in each location waiting for the customer to pick up the order (e.g., waiting in the lobby of the customer's condominium).

The data group at Uber wishes to consider routes that impose a **limit "W" on the maximum average waiting time** of the orders assigned to the driver. For example, consider again the same illustrative route from Part 1, now with the time each order is expected to be released under the restaurant icons.



Assume that the driver's velocity is 20 km/h (considering traffic and stopping lights, for example). The customer waiting time for the first order that is picked up in the route (i.e., R_1) is 15 minutes: the order is ready at 7:30 pm, the same time it is picked up, and it takes 15 minutes for the driver to arrive at N_1 (at 7:45 pm). The waiting time for the second order is 68 minutes: the driver leaves N_1 at 7:50 pm, picks the order at 8:02 pm in R_2, and arrives at N_2 in 8:08 pm. The average waiting time is (15+68)/2 = 41.5 minutes.

The solution is feasible for W = 60 minutes, but infeasible for W = 30 minutes, for example.

We still wish to minimize total distance traversed. Augment your model to incorporate this time constraint, evaluating your solution on the instances in *part2_ordersA.csv*.

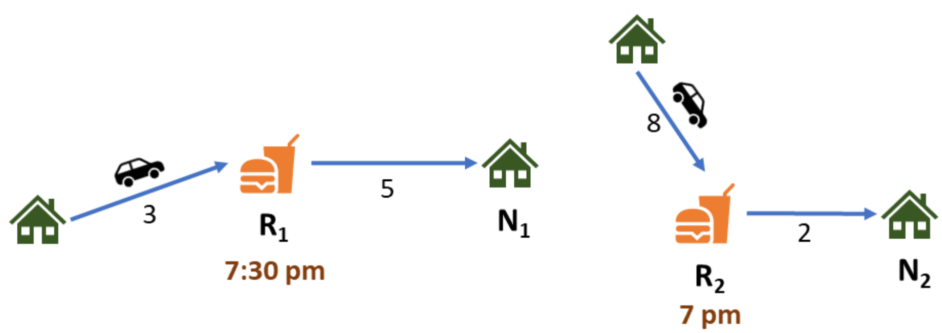
For this part, assume that the driver also starts at **Downtown Toronto (Rosedale)** and has an average velocity of **40 km/h**. Report the average waiting time for all your solutions.

Part III – The more the merrier

The routes calculated in Parts I and II help Uber Eats in estimating routing costs. The actual decision problem, however, is assigning orders to drivers based on such estimates.

Generalize your model from Part II to **multiple drivers**, that is, your model assigns orders to drivers as well as route them. Each driver has their own velocity based on their modal (walk, bicycle, or driving) and starting location described in the file **part3_drivers.csv**. Consider the average waiting time constraint from Part II (not your novel metric).

The objective is to minimize the **total distance traversed by all drivers**. For example, the solution below assigns the two orders from the previous examples to two distinct drivers, giving a total distance of 8 (first vehicle) + 10 (second vehicle) = 18 km.



Test your model on the instance *part3_small.csv*, collecting managerial insights from the solutions you obtained.

Recall to report always the total distance and average time in your report.

Data and further notes

You will also receive the file **regions.csv** with the geodata of the neighborhoods, and the file **distances.csv** with the distance in kilometers between two distinct neighborhoods. If you wish, you can use the Toronto geodata in this link (in GeoJSON) to plot your route in the map for visualization purposes.

It is extremely important that you build your models **incrementally**, always testing and inspecting solutions as much as possible. The models are not trivial. Please avoid implementing all constraints at once without testing and inspecting each portion of your model at a time.

The problems, even if small, are not easy to solve. If the runtime is taking too long, you can set a time limit with the code

```
model.solve(pulp.PULP_CBC_CMD(timeLimit=120))
```

where “120” represents the limit in seconds (you can use other numbers) and “model” is your PuLP formulation. The solver will halt at the time limit provided above with the best possible solution it could find during that time (if any). The total time elapsed after a solve() call can be obtained through the code

```
print("Time elapsed:", model.solutionTime)
```

All instances here (except *part3_large.csv*) are expected to be solved in less than two minutes. For the single-driver cases, taking more than 10 minutes could be indicative of problems in the formulation (possibly too many constraints). However, you may have different experiences based on your computer configuration. The solutions accepted in this case must be obtained within at least 10 minutes (600s).

Note: the solution is only optimal if the solver finishes **before** the time limit, even if PuLP says “optimal.”

Evaluation

Your report will be given a raw score out of 100 points using the following scheme:

- *Models* [40 points]: If the inventory models you designed are sound and adequate to the problem.
- *Analysis* [30 points]: if the analysis is sound and comprehensive, addressing the questions posed by the agency.
- *Implementation* [30 points]: If the implementation of your models is correct and free of errors or bugs. It must also be well organized and readable.