Computational Optimization
Homework 6

Nedialko B. Dimitrov

By completing this homework assignment you will learn about:

- Implementing complex custom optimization, given a journal publication of the algorithm.
- Data pre-processing and post-processing.

In this homework, you will write code that routes a truck driver around the restaurants of Austin, TX. To do this, you'll have to implement a custom algorithm for the Elementary Shortest Path with Resource Constraints problem. This custom optimization algorithm significantly out-performs the equivalent integer program for the same problem, and is described in a paper published in 2004 by Feillet, Dejax, Gendreau, and Gueguen called "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems" appearing in *Networks*, 44(3), 216-229. In implementing this algorithm, you will get practice in implementing an algorithm described in a journal publication – a key part of both research, and potential future problem solving. In addition, as with all algorithms, to make the algorithm useful, you'll have to do some data pre-processing, and solution post-processing.

1 Input Data

Download the file `hw06_files.zip`. The only difference in this data and the one for the previous homework are some extra columns in the `addresses.csv` file. The columns are `['ShortHand', 'LoadingTime','Value','Volume','TimeWindowStart','TimeWindowEnd']`. The **'ShortHand'** column gives a short description of the location. For each restaurant, the remaining columns describe the time required to pick up goods at the restaurant, the money earned by picking up the goods at the restaurant, the volume of the goods, and the time window in which we are allowed to pick up goods. For the location **'ETC'** the time window specifies the starting time of our truck driver's day, and the ending time of his day.

In addition to the data in `addresses.csv`, you will have to pre-process the Austin network, to collapse it to a network that only includes the restaurants and **'ETC'**, but has the correct driving times between all locations. You can use your code from the previous homework to help you do this. You will have to do a number of other pre-processing steps to create meaningful data input for the algorithm. While we sketched these other pre-processing steps in class, part of the assignment is to figure out these steps.

2 Implement the Algorithm

Implement the ESPRC algorithm, as described in the paper. In doing this, organize your code. Ask yourself the following design questions: "What kinds of objects do I need? What operations

do I need to do on those objects? How can I use the objects I create to implement this algorithm?" Likely you will have to edit, and re-edit your design as you implement... that's pretty typical.

3 Produce and Visualize Results

Produce routes that maximize revenue for the following three conditions. All the routes should start at **'ETC'** and end at **'ETC'**.

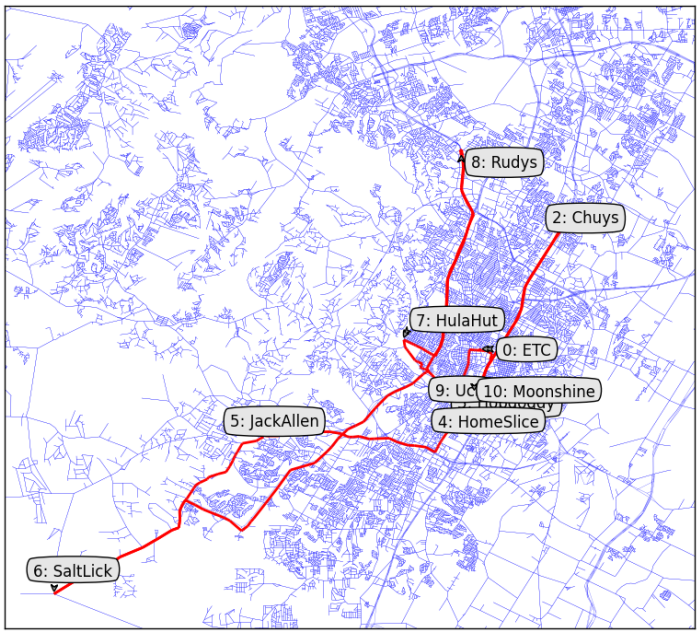
- A truck driver working 9am to noon, with a truck volume of 50. To help you debug, after pre-processing, my implementation of the algorithm takes about 108 ms to solve this instance, and I have about 19 labels on the final node. The best route has a value of 13.
- A truck driver working 8am to 5pm, with a truck volume of 100. To help you debug, after pre-processing, my implementation takes about 5.4 sec to solve this instance, with about 256 labels on the final node. The best route has a value of 29.
- A truck driver working 7am to 6pm, with a truck volume of 250. This instance takes about 15.6 sec, with 433 labels on the final node, and a route value of 38.

Write code to visualize and output the result. In particular, given a route, plot the route on a map of Austin... similar to Figure 1. In addition to the map, produce a `schedule.csv` file that has the driver's schedule. In general, there may be multiple optimal routes – however, for the second instance above, there is a unique route displayed the map in this writeup, and the example `schedule.csv` below.

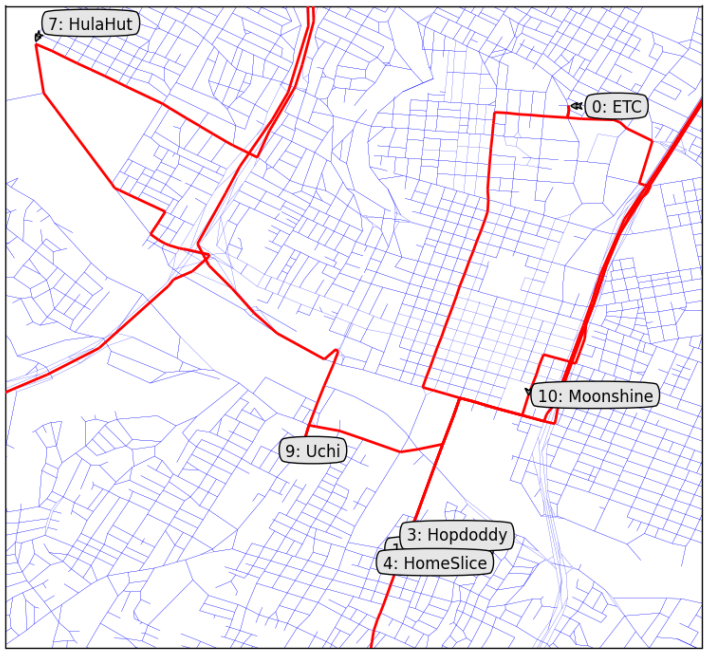
	ShortHand	ArriveTime	LeaveTime	ArriveVolume
0	ETC	7	7	0
1	EnotecaV	8	8.5	0
2	Chuys	9	9.5	5
3	Hopdoddy	9.6618	10.1618	22
4	HomeSlice	10.1632	10.6632	37
5	JackAllen	11	11.5	44
6	SaltLick	11.8570	12.3570	56
7	HulaHut	13	13.75	77
8	Rudys	14	14.5	83
9	Uchi	14.70274	15.2027	87
10	Moonshine	15.24428	15.7442	90
11	end	15.7973	15.7973	100

4 Final Thoughts

In our discussions in class, we created a model that arrives at each restaurant in the given time window, but might leave *outside* the time window, after loading. How would you change the model so that the driver has to both arrive and leave within the specified time window?



(a) Best route for second instance



(b) Best route for second instance, zoom

Figure 1: Route for the second instance (8am to 5pm, volume 100). There may be multiple optimal solutions, the above pictures are for just one of them.

What to turn in: For this assignment, turn in four things:

1. Your Python code in a file called `hw06.py`
2. A plot of an optimal route for the second instance (8am to 5pm, volume 100).
3. A `schedule.csv` for the second instance.
4. A short (less than one minute) video explanation of your code.

Key take-aways: From this assignment, you should take-away these things:

1. You had to do a significant ammount of pre-processing and post-processing of the data. These important steps surround the optimization algorithm in any useful piece of software. Sometimes, these parts are as complex as the algorithm itself.
2. Now you have experience implementing a complex custom optimization algorithm described in a publication. In implementing the algorithm, you gain a much clearer understanding of how and why it works. You will likely have to repeat this process many times either for your thesis or for a future employer. The understanding you gain of the algorithm might give you great ideas on how to improve it, both practically and theoretically.